Beauto Racer 専用ソフトウェア

Beauto Builder R

ビュート ビルダー アール

取扱説明書

ヴイストン株式会社



目次

1.はじめに

この度は、「Beauto Racer」（以下、「**ロボット**」及び「**ロボット本体**」と記述）をお買い上げいただきありがとうございます。Beauto Builder R（以下、「**本ソフトウェア**」と記述）は、ロボットを PC と接続してロボットのプログラムを作成するためのソフトウェアです。「Beauto Builder R 取扱説明書」（以下、「**本書**」と記述）は、本ソフトウェアの操作方法について説明した資料です。説明の手順は、プログラミング課題を例示し、実際にロボットを動かしながら実習することを想定しているため、なるべく最初から順番に読み進めてください。なお、初めてプログラムを学習される方は、本書より簡単にプログラミングが学習できる「**BeautoBuliderR プログラム学習の手引き**」を先にお読みください。

1-1.ソフトウェアのインストール

まず、本ソフトウェアを PC にインストールします。本ソフトウェアは、下記の環境を満たす PC で動作します。

- **OS：Windows2000/XP/Vista**
- **CPU：Pentium-Ⅲ以降（1GHz 以上推奨）**
- **RAM：128MB**
- **インターフェース：USB ポート**
- **画面サイズ：XGA(1024×768)以上**
- **シミュレータ機能を利用する場合は、DirectX9.0b 以降のインストールが必要**

本製品に付属 CD-ROM をお持ちの方は PC に CD をセットして、マイコンピュータから CD のドライブを開いてください。CD には「BeautoBuilderR」という名前のフォルダが入っているので、フォルダごと PC の好きな場所(デスクトップなど)にコピーしてください。これでインストールは完了です。

また、付属 CD-ROM をお持ちでない方は、下記のユーザサポートページにアクセスし、最新版の本ソフトウェアをダウンロードしてください。

●Beauto Racer サポートページ URL
http://www.vstone.co.jp/top/products/robot/beauto/rdownload.html

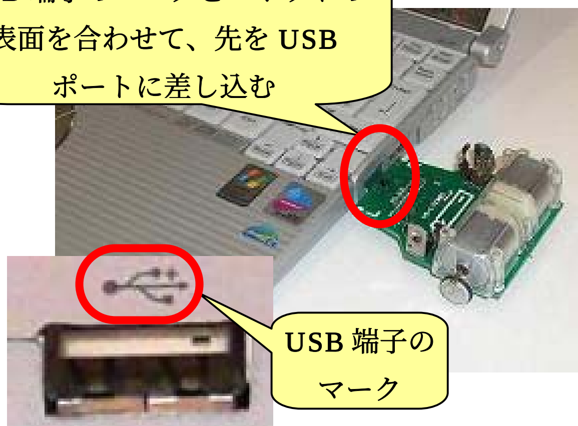
ダウンロードしたファイルは ZIP 形式で圧縮されているので解凍してください。解凍すると「BeautoBuilderR」というフォルダができるので、そのフォルダを丸ごと PC の好きな場所（デスクトップなど）にコピーしてください。



1-2.PC とロボットとの接続・認識

本ソフトウェアのインストールが完了したら、ロボットを PC に接続して認識させます。まず、**ロボット本体の電源スイッチを OFF にします。**続いて **USB 端子のマーク（下図参照）のある方にロボットの表面を合わせて、**先端を PC の USB 端子に差し込んでください。

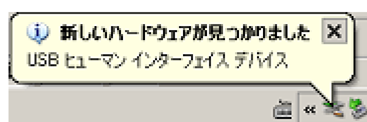
ちなみに、市販の「USB 端子延長ケーブル」を使用すると、ロボットを PC に直接接続せずに、ケーブルを通じて接続することができます。



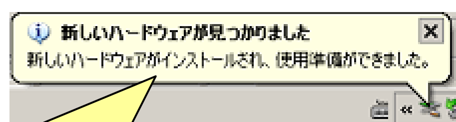


ロボットを PC に接続する場合、必ずロボットの電源スイッチを OFF にしてください。
電源スイッチが ON になっていると、通信が途切れたり勝手にモータが動いたりする場合があります。

はじめて PC とロボットを接続した場合、PC の画面に下記のようなフキダシが表示されます。画面に「新しいハードウェアがインストールされ、使用準備ができました」というメッセージが表示されればロボットの認識は完了です。

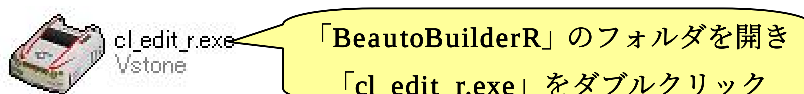




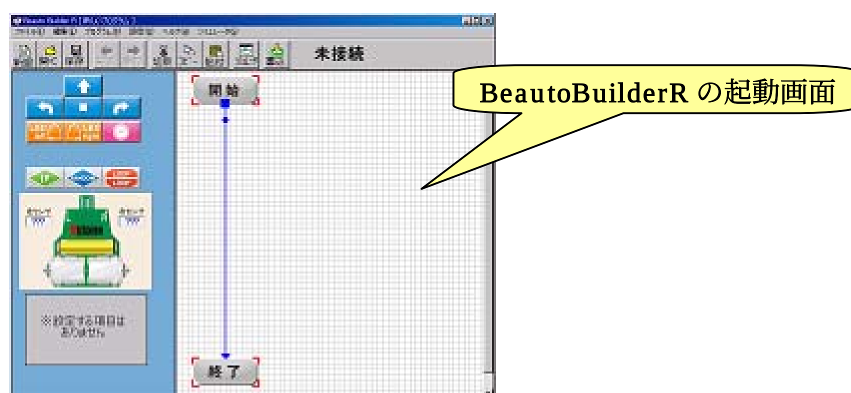


次に PC とロボットを接続したときには、フキダシを表示せずロボットを自動的に認識します。

それでは、ここまでの作業が正しく行なわれたかを確認するため、本ソフトウェアを起動してロボットと通信させてみたいと思います。まず、PC にコピーした「BeautoBuilderR」のフォルダを開いてください。フォルダを開くと中に「**cl_edit_r.exe**」という実行ファイルが入っているので、ダブルクリックして実行してください。

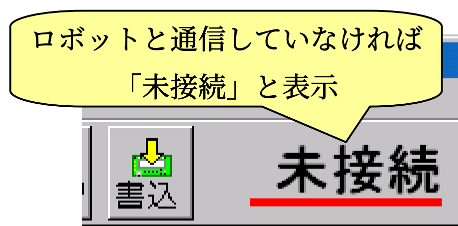


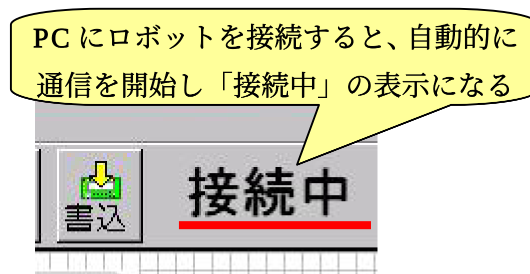
クリックすると本ソフトウェアが起動し、画面に下記のウインドウを表示します。



※本ソフトウェアを実行すると「gdiplus.dll が指定したフォルダに見つかりません・・・」というエラーが表示され実行できない場合は、「BeautoBuilderR」のフォルダに入っている「gdiplus」というフォルダを開き、その中にある「gdiplus.dll」というファイルを「cl_edit_r.exe」と同じフォルダ（「BeautoBuilderR」のフォルダ）にコピーしてください。

ウインドウの右上には現在のロボットとの通信状況が文字で表示されます。ロボットが PC に接続されていないなど通信をしていない状態では、「**未接続**」と表示されます。また、PC とロボットを接続していると自動的に通信を開始し、ウインドウ右上の文字が「**接続中**」に変わります。





また、画面左側の「**左センサ**」「**右センサ**」の部分に、現在のロボットのセンサ値が表示されます。



もし、PC にロボットを接続しても「接続中」にならない場合は、次の原因が考えられます。それぞれで説明している解決方法を実行してください。

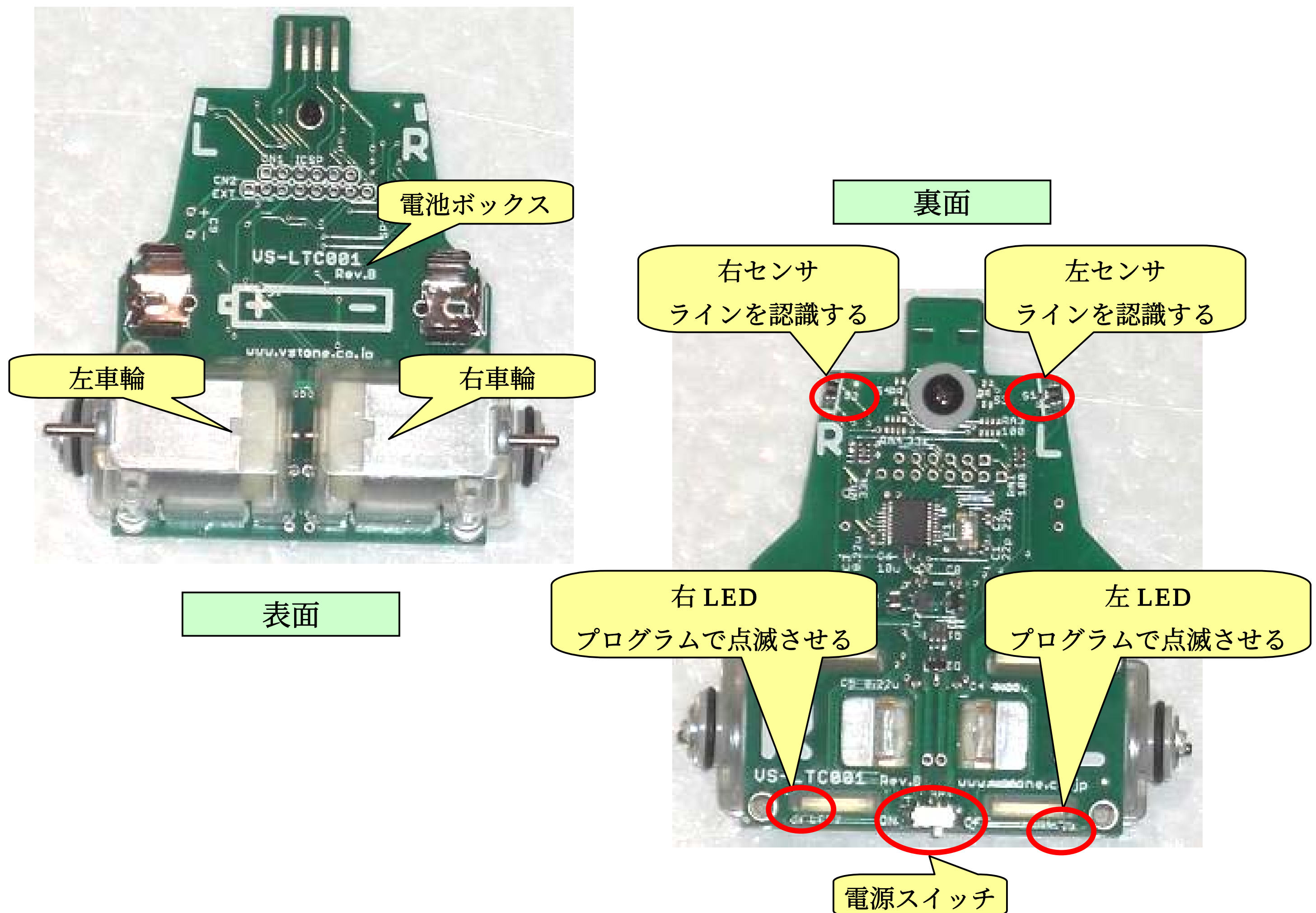
・ PC の USB 端子にロボットが奥までしっかり差し込まれていない、ロボットが斜めに差し込まれている
  - **ロボットを真っ直ぐ USB 端子の奥までしっかり差し込み、PC とロボットが正しく接続されるようにしてください**

・ ロボットの表面と裏面を逆にして差し込んでいる
  - **ロボットの裏表を合わせなおして USB 端子に差し込み、PC とロボットが正しく接続されるようにしてください**

・ PC にロボットを何度も抜き差ししている
  - **ロボットを何度も抜き差ししていると PC がロボットを認識するまでに若干時間がかかるようになる場合があります。この状態でも通信は正しく行なわれますが、気になる場合は一度 PC を再起動してください。**

1-3.ロボットのハードウェア説明

ロボット本体には、モータ・センサ・LED などのインターフェースが備わっています。それぞれの搭載場所や名称は下図の通りです。



2.単純なプログラムの作成

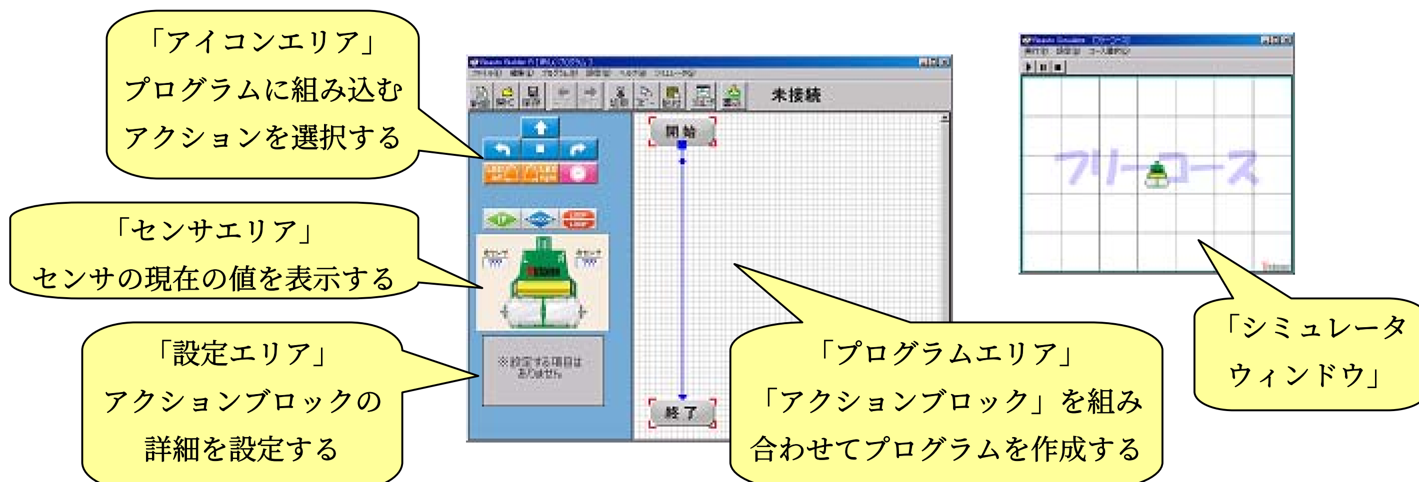
それでは、本ソフトウェアでのプログラミングについて説明したいと思います。まずは、本ソフトウェアの基本的な使い方や使用できる命令などの基本部分を、実際にプログラミングを進めながら説明していきます。

2-1.画面の説明

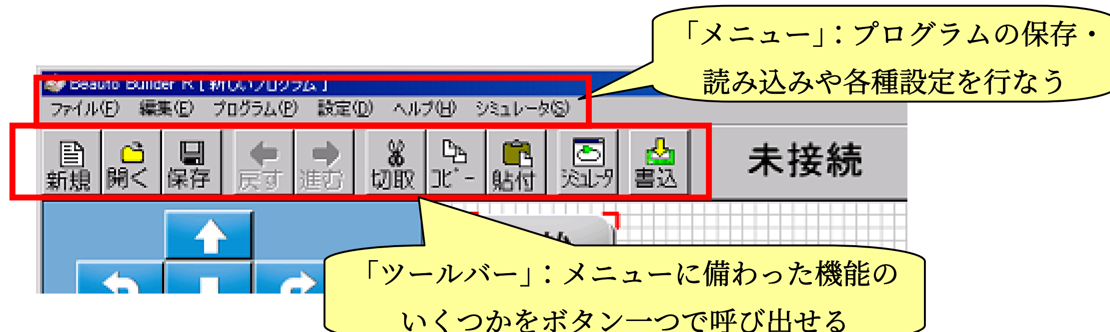
本ソフトウェアの画面は、左右に大きく分かれます。左側は大きく三つに分かれます。

画面左側は、上から順番に、プログラムに使用する「**アクション**」(命令) を選択する「**アイコンエリア**」、ロボットの現在のセンサ情報を表示する「**センサエリア**」、アクションの詳細を設定する「**設定エリア**」といいます。

画面右側はアクションを組み合わせてプログラムを作成する「**プログラムエリア**」といいます。また、シミュレータを利用する場合は、「**シミュレータウィンドウ**」を別に開きます。

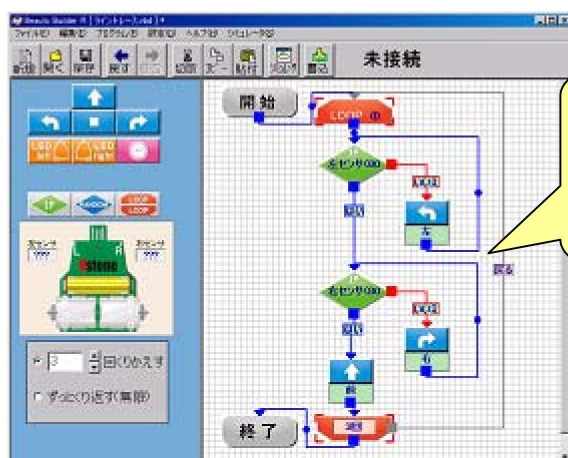


また、画面の上側には「メニュー」と「ツールバー」が備わっています。メニューはウインドウタイトルの真下の文字列で、プログラムの保存や読み込み、ロボットに関する設定などを行ないます。ツールバーはメニューの真下に横に並んでいるボタンを表し、メニューに備わっている機能のいくつかをボタン一つで呼び出せるようになっています。



画面右のプログラムエリアには、アクションが種類に応じて色分け・記号化した「**アクションブロック**」でわかりやすく表示されます。アクションブロックは青や赤の矢印でつながっており、この矢印がプログラムを実行する順番になります。プログラムは必ず「**開始**」のアクションブロックから始まり、「**終了**」のアクションブロックで終わります。

このように、本ソフトウェアではプログラムの目的に応じて、様々な種類のアクションブロックをフローチャートのように接続してプログラムを作成します。

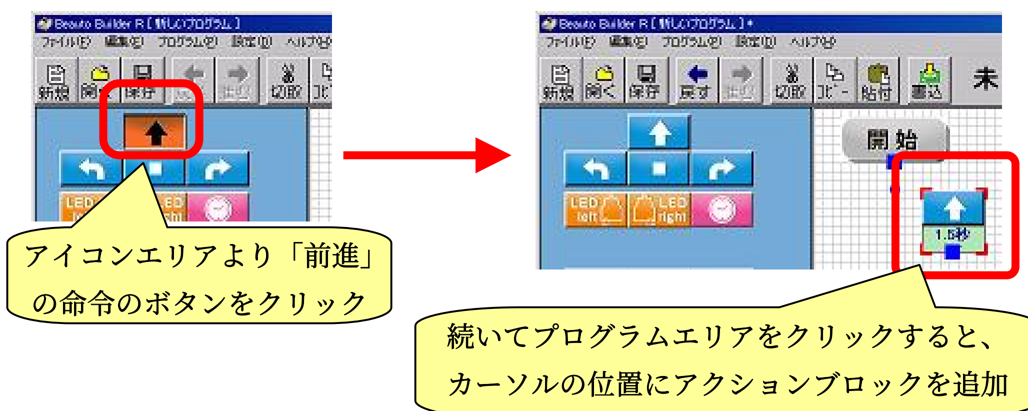


2-2.アクションブロックの追加

それでは、まずはロボットが前進するだけの最も簡単なプログラムを作成しましょう。

最初に、プログラムで使用するアクションブロック（命令）を、画面左のアイコンエリアから選択し、続いて画面右のプログラムエリアの好きな場所に追加します。

まず、アイコンエリアより上向きの矢印のボタン（下図参照）をクリックしてください。このボタンは「前進」の命令を表します。クリックするとボタンの色が変わります。続いて、プログラムエリアの好きな場所をマウスでクリックしてください。クリックすると、その場所にアイコンエリアで選択した命令が「アクションブロック」として追加されます。



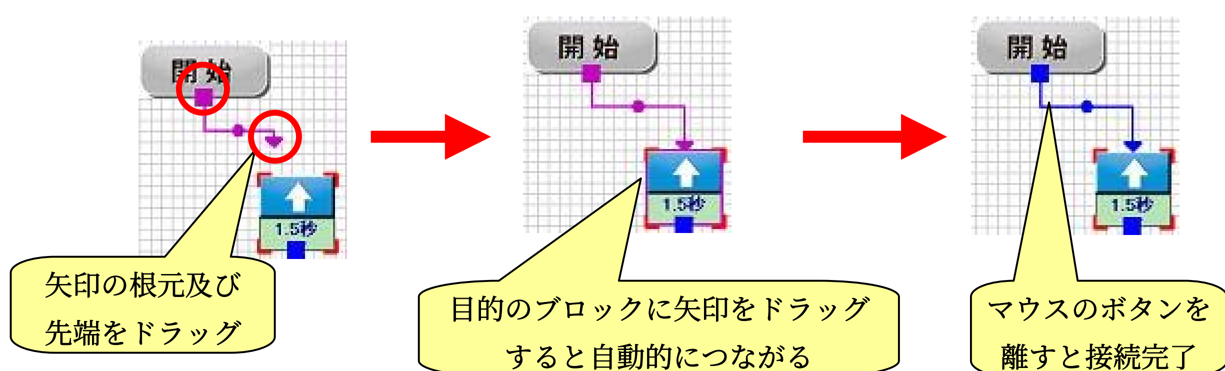
プログラムで使用する命令は、このように「**使用する命令の種類をアイコンエリアよりクリック**」→「**プログラムエリアの命令を挿入する場所をクリック**」という手順で追加します。

2-3.フローチャートの組み立て

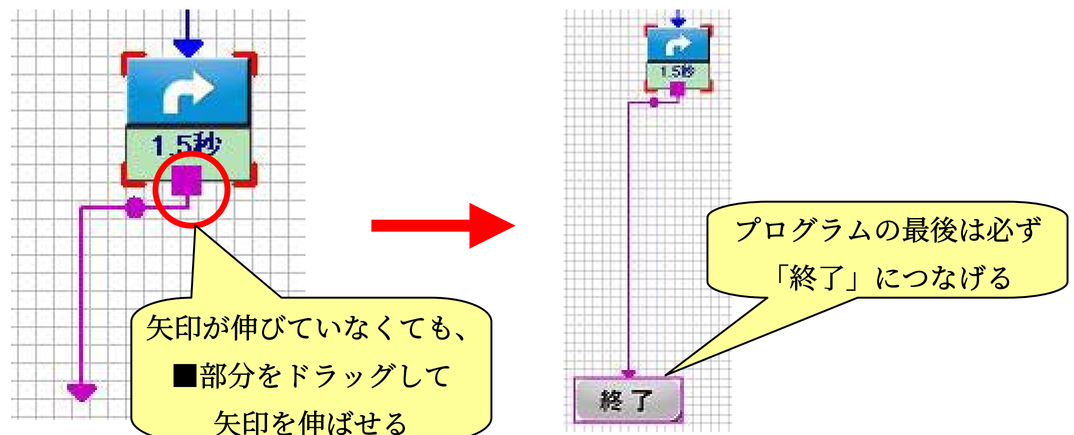
これで「ロボットに前進させる」命令がプログラムに追加されましたが、このままではプログラムを実行してもロボットは動きません。なぜなら、**プログラムの「開始」と「終了」の矢印の中に、追加した前進の命令が組み込まれていないためです。**追加した命令をロボットに実行させる場合は、それぞれのアクションブロックの矢印をつなぎ変えて、フローチャートの中に組み込む必要があります。それでは、矢印をつなぎ変えてみましょう。

アクションブロックの矢印をつなぎかえる場合は、**アクションブロックの矢印の根元、もしくは先端をクリックしてドラッグします。**矢印の根元の■部分、もしくは先端の▲部分をドラッグすると、**矢印の色が紫に変わり、先端の位置を自由に動かせるようになります。**そのまま**別のアクションブロックにマウスカーソルを重ねると、矢印が自動的につながります。**その状態で**マウスのボタンを離すと、矢印をつなぎかえることができます。**

では、まず下図のように「開始」のアクションブロックの矢印を「前進」のアクションブロックに接続してください。



これと同じように、「前進」のアクションブロックの矢印を「終了」のアクションブロックに接続してください。「前進」のアクションブロックの矢印は伸びていませんが、根元の■部分をクリックすれば矢印を伸ばせます。



このように、プログラムを作成する際は、**最後の命令の矢印を必ず「終了」のアクションブロックに接続してください。**

※「開始」のアクションブロックのみ、矢印を接続することができずすり抜けます。

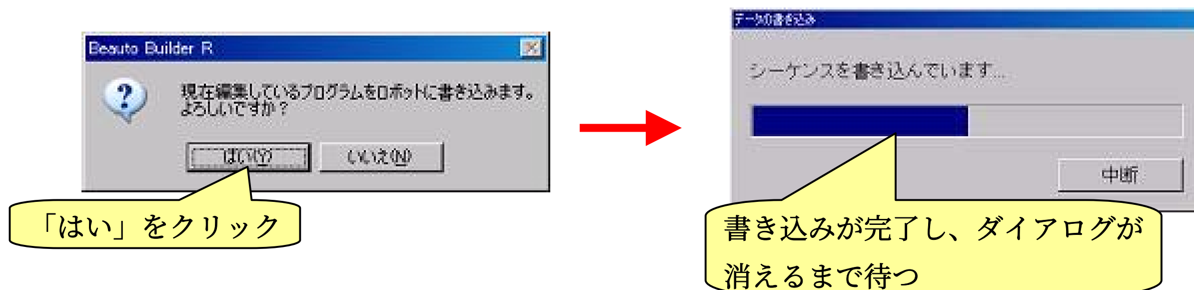
2-4.プログラムの書き込み・実行

それでは、作成したプログラムをロボットに書き込んで実行させてみましょう。まず、PC にロボットを接続して通信をさせてください。PC がロボットと通信中であることを確認したら、ボタンをクリックしてください。



※シミュレータでプログラムを実行する場合は、「**5.シミュレータについて**」を参照してください。

ボタンをクリックすると、本当に書き込みを行なうか確認のメッセージを表示するので「はい」をクリックしてください。クリックすると、画面にプログラムの書き込み状況を表すウインドウを表示して書き込みが始まります。書き込み状況を表すウインドウが消えたら書き込みが完了です。



※プログラムの書き込み中は、ロボットを PC から取り外したり、激しく動かすなど触らないようにしてください。

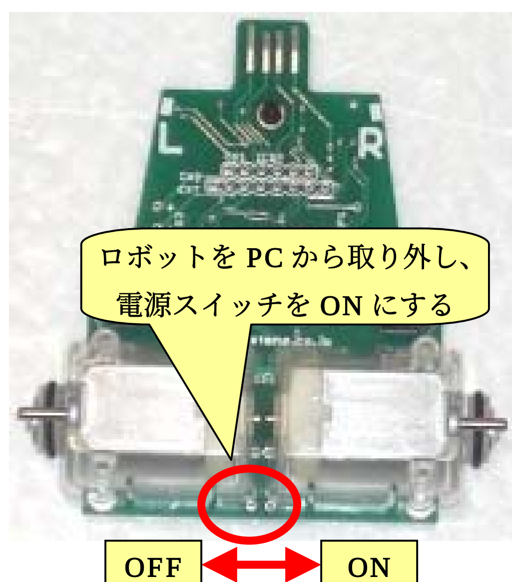
書き込み中にロボットとの通信が途切れるなどにより、正常にプログラムを書き込みできなかった場合、下記のメッセージが表示されます。この場合、「OK」をクリックしてメッセージを閉じ、もう一度正しい手順で書き込みを行ってください。



プログラムの書き込みに失敗する主な原因として、「書き込み中にロボットに触ったり動かしたりする」ことがあげられます。プログラムの書き込み中にロボットに触れると、通信が途中で途切れるなどして書き込みに失敗しやすくなります。**書き込み中はロボットに触れず、書き込みが終わるまでなるべくそっと置いておいてください。**

プログラムの書き込みが終わったら、プログラムを実行してみましょう。**ロボットは、PC から取り外した状態で電源スイッチを ON にすると、すぐにプログラムを開始します。**では、まずロボットを PC から取り外してください。そして、**ロボットが動いたときに机から落ちたり物にぶつかったりしないように、安全な場所にロボットを置き、**準備ができたら電源スイッチを ON にして手を離してください。

電源スイッチを ON にすると、作成したプログラムの通りロボットが約 1.5 秒前進して止まります。**最後までプログラムを実行するとロボットは停止するので、もう一度プログラムを開始する場合は、一度電源スイッチを OFF にして再び ON にしてください。**また、**途中でプログラムを止める場合は、電源スイッチを OFF にしてください。**



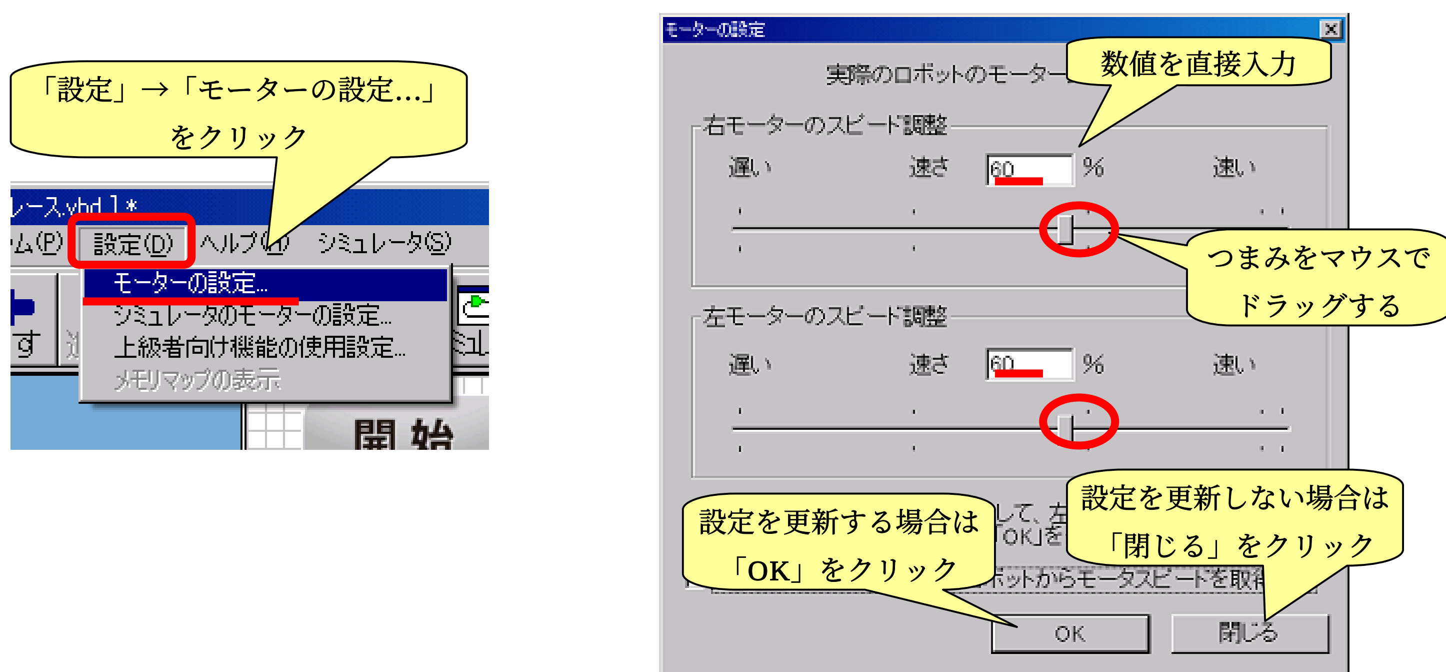
片方、もしくは両方のモーターが動かないなど、プログラムが正常に動作しない場合は、ロボットの組み立てに問題がある可能性があります。次の点について改めて確認してください。

- **モーターを表裏反対に取り付けている**
  - モーターの茶色い金具がロボット本体の基板とくっつく向きが正しいです。モーターを正しい向きに直して組み立ててください。
- **モーターの茶色い金具が裏の穴から飛び出している**
  - 組み立て時にモーターの金具を起こしすぎています。もう少し金具を寝かせて組み立ててください。
- **電池をプラス・マイナス逆に入れている**
  - ロボット本体の基板に描かれている図の通りに、正しい向きで電池を入れて下さい
- **モーターホルダーを指で押し込むとモーターが正しく動く**
  - モーターホルダーの押し込みが足りない、もしくは、モーターの金具を充分起こしていない可能性があります。組み立て説明書を参考にモーターの金具を 30 度くらいまで起こし、しっかりモーターホルダーを押し込んでください。

2-5.モータースピードの調整

プログラムを実行したときに、「**ロボットが真っ直ぐ進まず、右・左に微妙に曲がる**」という人もいると思います。これは、部品の個体差やロボットを動かす地面などの影響によるものです。このプログラムでは、ロボットが真っ直ぐ走ることが理想なので、**本ソフトウェアより左右のモーターのスピードを調整して、ロボットが真っ直ぐ走るようにしたいと思います。**

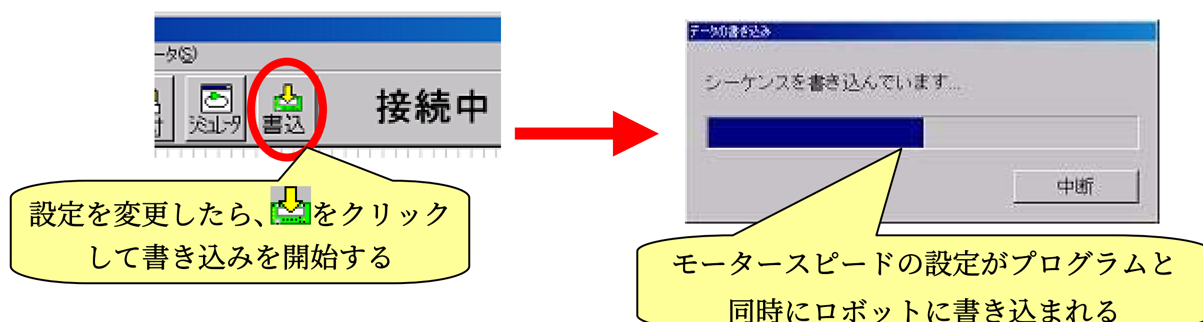
まず、画面上部のメニューより「設定」→「モーターの設定…」をクリックしてください。クリックすると、設定を行なうダイアログを開きます。



モータースピードは、左右のモーターで個別に設定できます。設定値は 0~100%の範囲で、数値が大きいほどモーターのスピードが速くなります。数値は、スライダーのつまみをマウスでドラッグしたり、キーボードから数値を直接したりして設定します。設定を変更したら、「OK」をクリックすることで決定できます。また、変更をやめる場合は「閉じる」をクリックします。

また、シミュレータのロボットのモータスピードは、これとは別に行ないます。詳しくは「**シミュレータについて**」をご参照ください。

変更したモータースピードをロボットに反映させる場合は、ロボットに数値を書き込む必要があります。モータースピードの設定は、プログラムをロボットに書き込む際に併せて行なわれます。モータースピードの設定を変更した場合は、ロボットにプログラムを書き込んでください。

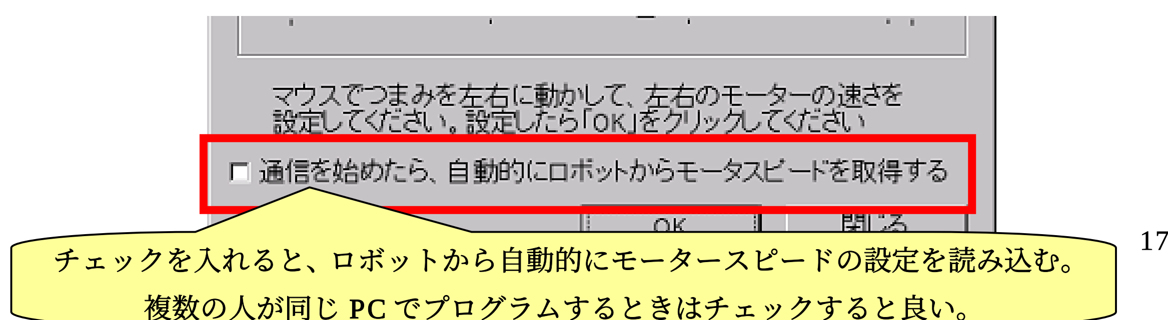


書き込み終わったら、改めてプログラムを実行し、モータースピードが正しく調整されているか確認してください。まだ調整が足りない場合は、本ソフトウェアから調整しなおしてロボットに数値を書き込んでください。

なお、はじめに説明したとおり、モータースピードは部品の個体差やロボットを動かす地面の影響で変化するため、**部品を交換したり別の場所でロボットを動かす場合は、再びずれが発生する可能性があります。**また、**モーターのスピードが極端に速い・遅い場合はロボットがプログラムどおりにうまく動かない場合があります。**その場合は、改めて調整作業を行なってください。

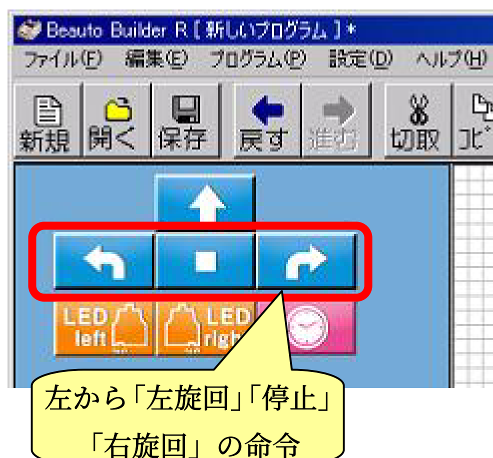
モータースピードの調整をしなくても、ロボットがはじめから真っ直ぐ動く場合は、特に調整する必要はありません。ただ、調整の方法を覚えるためにも、一度設定を変更してロボットの動きが変わることを確認してみましょう。

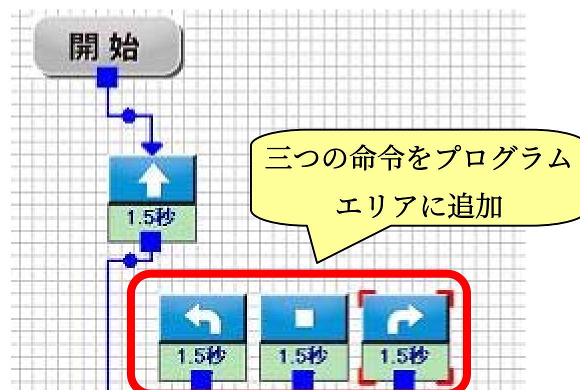
また、ダイアログ下の「通信を始めたら、自動的にロボットからモータスピードを取得する」の項目にチェックを入れると、PC とロボットが通信を始めたときに、接続したロボットからモータースピードの設定を自動的に読み込むようになります。様々な人が使う PC などでロボットのプログラムを作成する場合は、その PC でこの項目にチェックを入れると、他の人のモータースピード設定を、間違えて自分のロボットに書き込んでしまう可能性が低くなります。ロボットをお使いの環境に合わせて設定をお選びください。



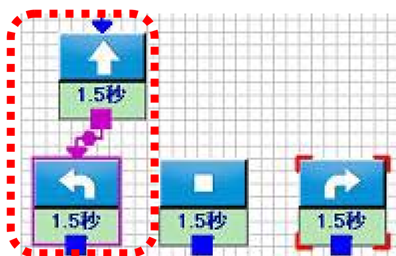
2-6.他のアクションブロックの追加

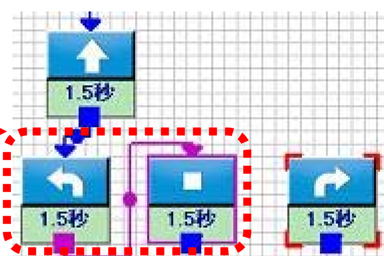
それでは、次に複数の命令を含んだプログラムを作成してみましょう。先ほどのプログラムに、「停止」「右旋回」「左旋回」の命令を追加してください。

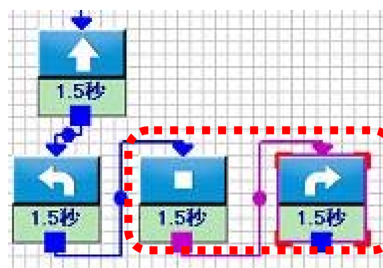


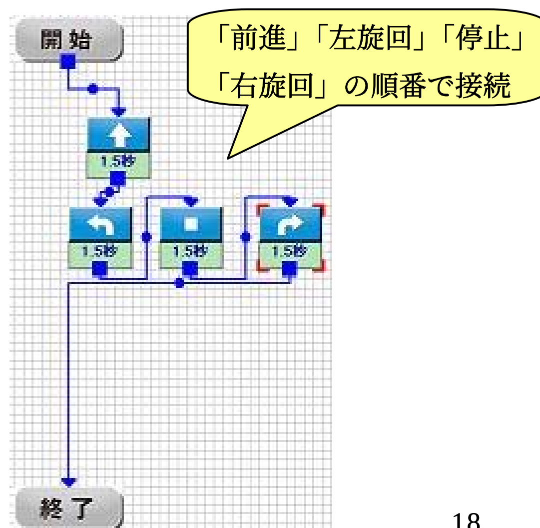


命令をプログラムエリアに追加したら、先ほどと同じようのアクションブロックの矢印をつないでプログラムを組み立てます。それでは、「前進」→「左旋回」→「停止」→「右旋回」の順番で命令を接続してください。



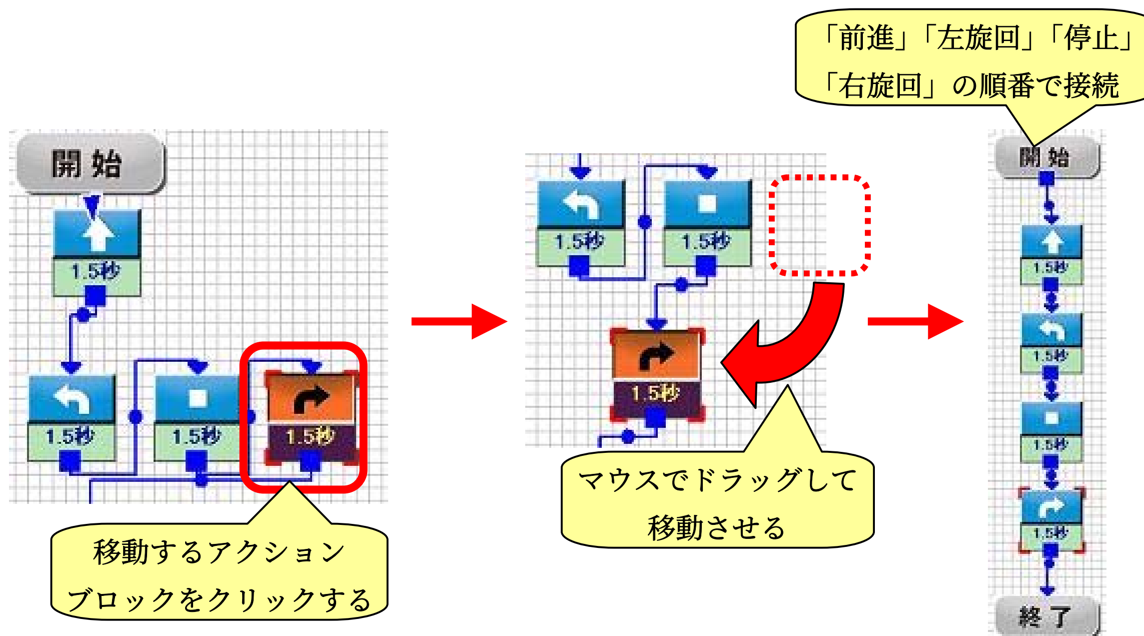




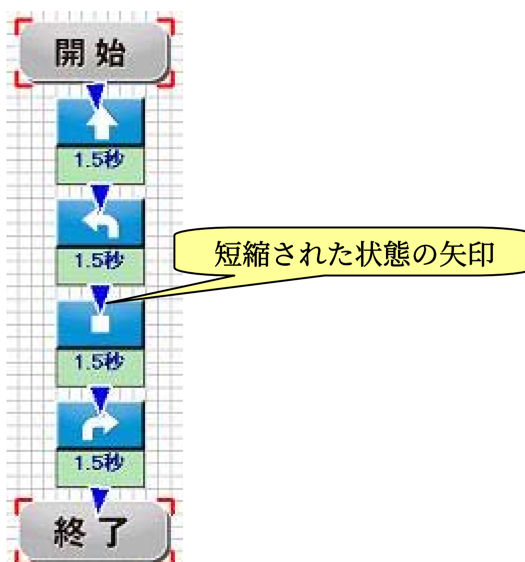


プログラムを作成したら、ロボットに書き込んで実行してみてください。実行すると、ロボットがプログラムの通りに「前進」→「左旋回」→「停止」→「右旋回」の順番で動作します。

プログラムエリアにアクションブロックが増えると、配置によってプログラムが見づらくなってしまうので、**マウスでアクションブロックをドラッグして見やすい位置に移動し**ましょう。一般的なフローチャートでは、プログラムの流れに従って命令を縦に記述するので、その順番の通りにブロックを並べ替えてみましょう。



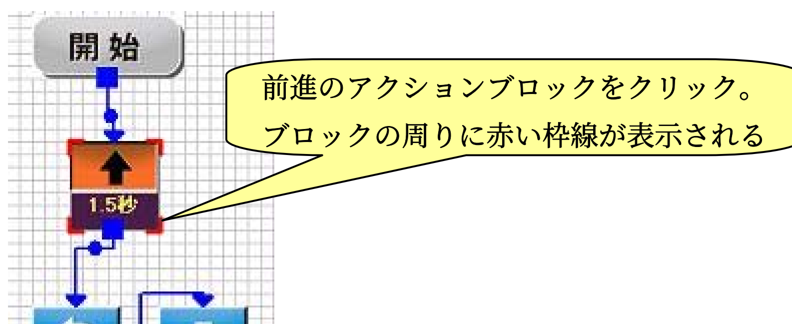
また、矢印をつないだアクションブロック同士が近づくと、下図のように矢印が短縮して表示されます。



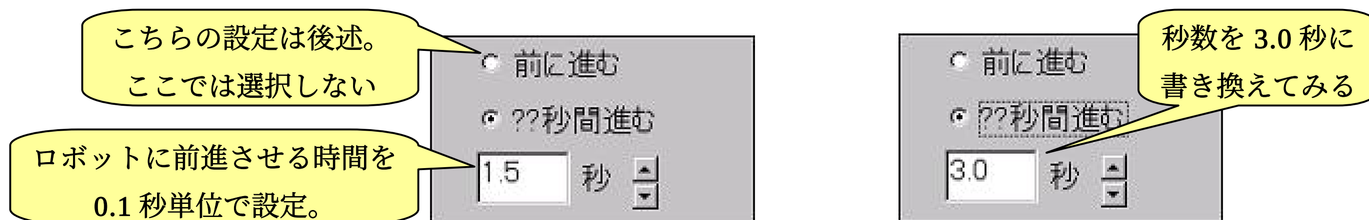
2-7.命令の詳細設定

これまでのプログラムでは、アイコンエリアから追加した命令をそのままプログラムに組み込みましたが、**各命令は「車輪を動かす時間を変える」など、更に細かい設定ができる**ようになっています。それでは、プログラムに追加された命令の設定をそれぞれ変更してみましょう。

まず、「前進」のアクションブロックをマウスでクリックしてください。**クリックするとブロックの色が変わるとともに、またブロックの周りに赤い枠線が表示**されます。



続いて、**画面左下の「設定エリア」**を確認してください。設定エリアは、命令の細かい設定を変更する部分です。「前進」のアクションブロックをクリックすると、このエリアの内容が下図のように変わります。



ここで表示されている「1.5 秒」の数値は、**ロボットに何秒間前進させるか**の設定を表しています。この数値を書き換えることで、ロボットに現在のプログラムよりも長い（もしくは短い）時間前進させることができます。秒数は 0.0～25.5 秒の範囲で 0.1 秒ずつ設定できます。また、数値の設定はキーボードから入力したり、右横のスピンボタンをクリックしたりして変更できます。それでは、最初の設定である「1.5 秒」から「3.0 秒」に数値を変更してください。

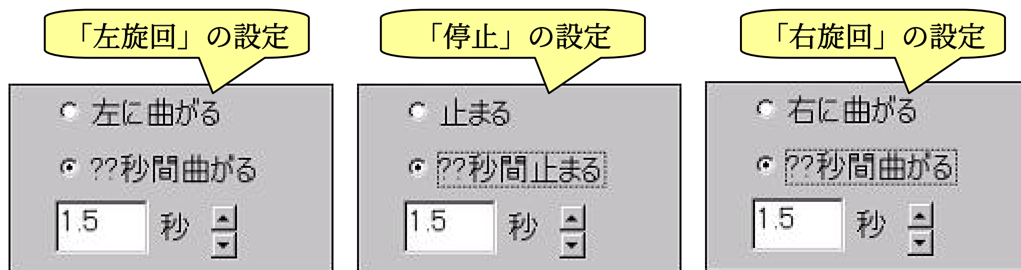
なお、「前に進む」の設定については、扱いが難しいためここでは説明を省きます。詳しくは後述の「センサを使うプログラムの作成」を参照してください。

数値を変更したら、再びロボットにプログラムを書き込んで実行し、前進する時間が長くなったかどうか確認してください。

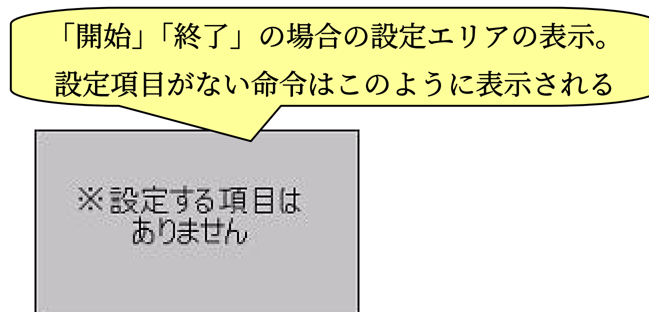


プログラムを実行して「前進」の時間が変わったことを確認できたら、「右旋回」「停止」などの他の命令についても、プログラムエリアからクリックして設定エリアの内容を確認してみましょう。

「右旋回」「左旋回」「停止」の命令については、先ほどの「前進」と同じようにモーターを動かす（もしくは止める）時間を設定できます。



一方、「開始」「終了」については、特に変更できる設定項目が無いため、設定エリアには下図のように表示されます。



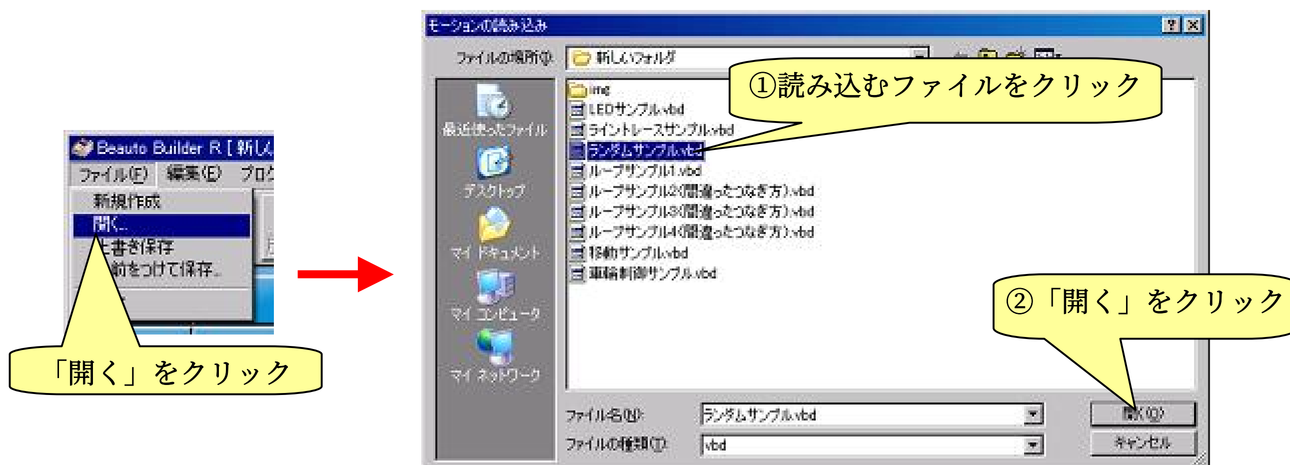
2-8.プログラムの保存・読み込み

　本ソフトウェアで作成したプログラムは、ファイルに保存したりファイルから読み込んだりすることができます。プログラムをファイルに保存する場合は、次の手順で行います。

　プログラムをファイルに保存する場合は、メニューより「ファイル」→「名前をつけて保存」をクリックしてください。クリックすると右下図のような画面が表示されるので、プログラムのファイル名を入力して「保存」をクリックしてください。



　また、ファイルからプログラムを読み込む場合は、メニューの「ファイル」→「開く」をクリックします。クリックすると右下図のような画面を開くので、読み込むファイル名を入力して「開く」をクリックしてください。

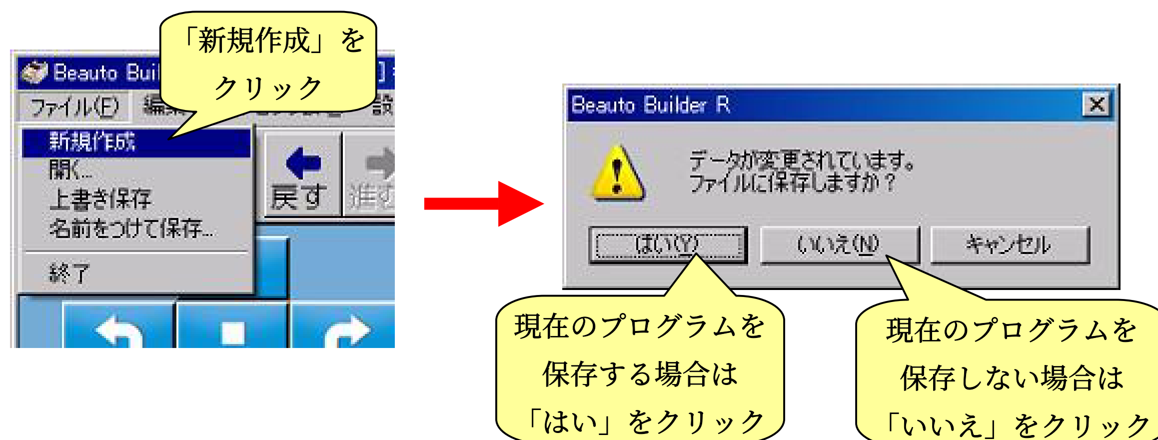


ちなみに、画面上部のツールバーより💾ボタンをクリックすることでもファイルを保存できます。また、📂ボタンをクリックするとファイルからプログラムを読み込みます。



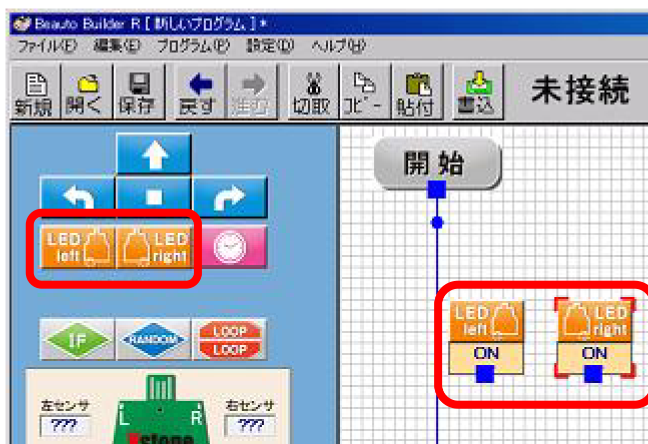
2-9.LED・ウエイトの命令

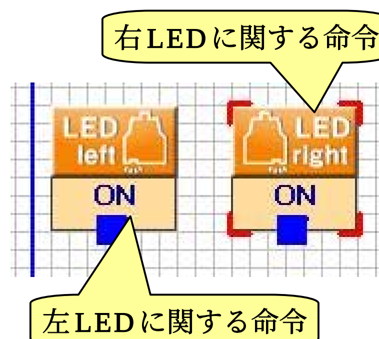
それでは、次にモーターを動かす以外の命令を使ってみたいと思います。その前に、現在作成しているプログラムを消して新しいプログラムの作成を始めます。メニューより「ファイル」→「新規作成」をクリックしてください。このとき、現在作成しているプログラムをファイルに保存していないと、右下図のメッセージが表示されます。現在のプログラムをファイルに保存する場合は「はい」を、保存しない場合は「いいえ」をクリックしてください。



また、ツールバーより📄ボタンをクリックしても、プログラムの新規作成を行なうことができます。

プログラムの新規作成を行なったら、アイコンエリアより下図の二つの命令を追加してください。これらはロボット本体の LED を操作する命令です。「LED left」と書かれた方が左 LED、「LED right」と書かれた方が右 LED をそれぞれ操作します。





設定エリアでは、LED の命令は下図のように表示されます。設定エリアでは、LED の点灯・消灯について三つの項目から選択します。「点ける(ON)」は LED を点け、「消す(OFF)」は LED を消します。また、「??秒間点ける」は、指定した秒数だけ LED を点け、時間が来たら LED を消します。

LED を点ける。LED は点けっぱなしで次の命令に進む

LED を消す。LED は消しっぱなしで次の命令に進む

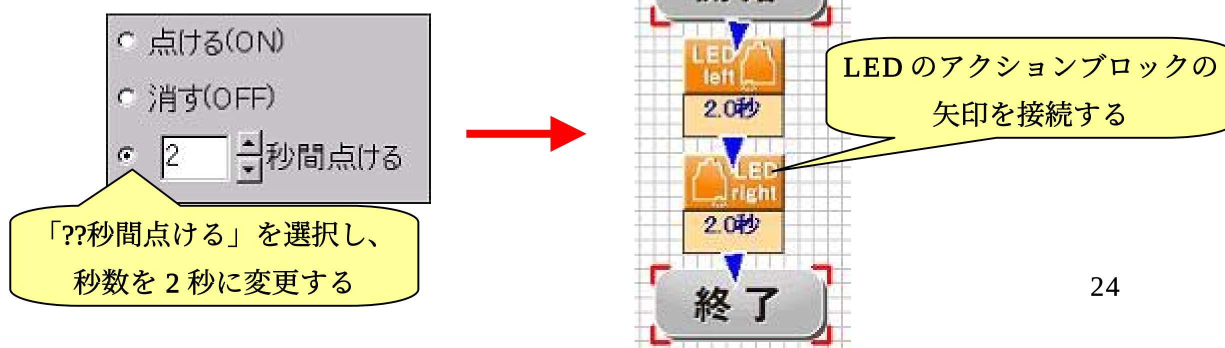
点ける(ON)

消す(OFF)

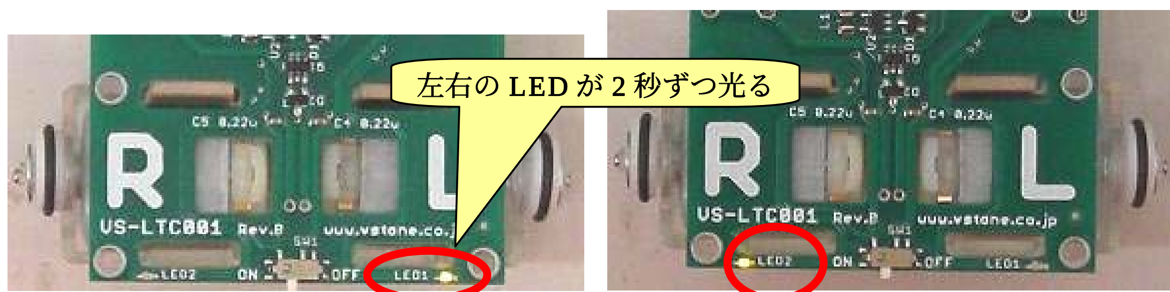
1.0 秒間点ける

指定時間だけ LED を点ける。時間が来たら LED を消して次の命令に進む

それでは、まず左右の LED を 2 秒ずつ光らせるプログラムを作成してみましょう。二つの LED の命令について、現在の「点ける（ON)」から「??秒間点ける」に変更し、秒数を 2 秒に設定してください。設定エリアでの変更ができたら、プログラムエリアより二つの LED のアクションブロックを接続してください。

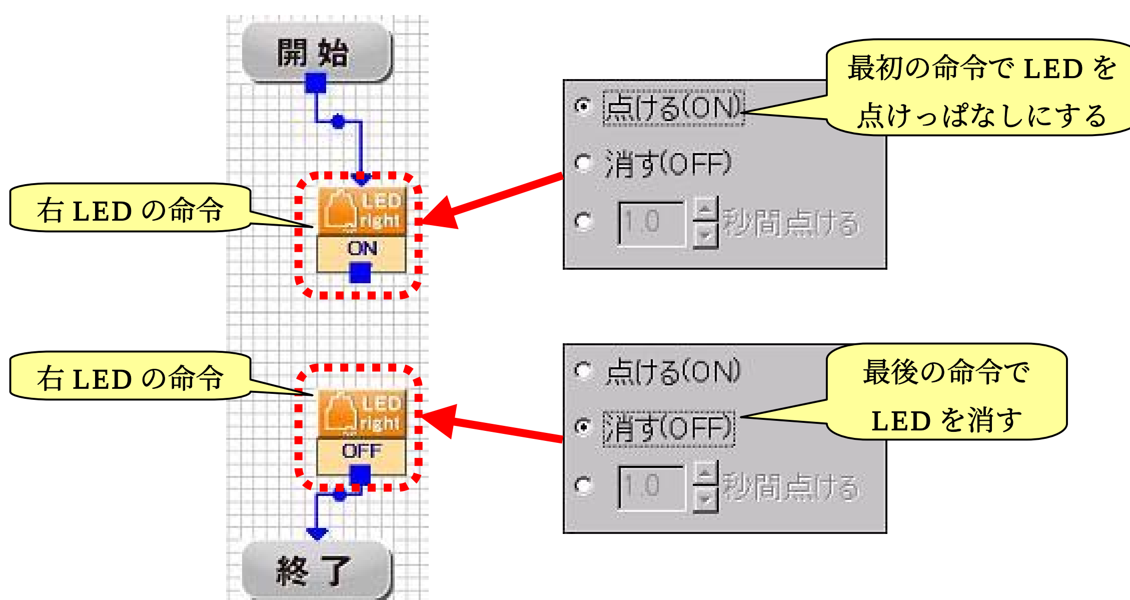


プログラムを作成したら、ロボットに書き込んで実行してみましょう。実行の際にはロボットを裏返し、LED が正しく光るか確認します。



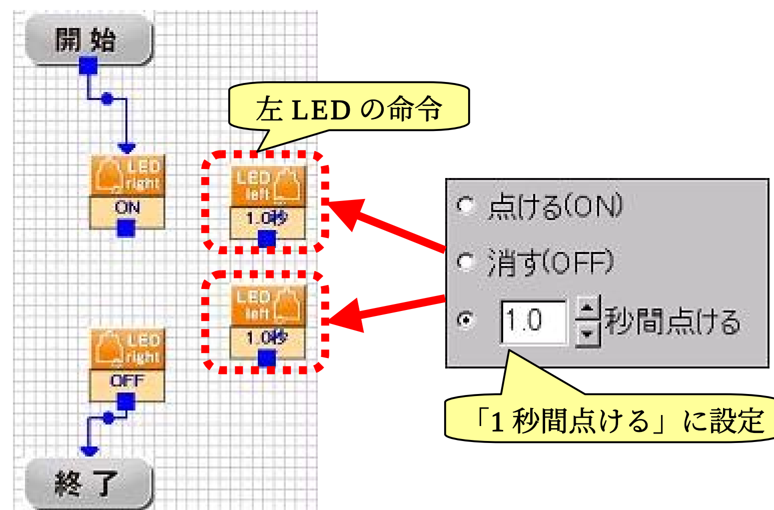
それでは、次に「**右の LED を点けっぱなしにして、左の LED を 1 秒間隔で点滅させる**」というプログラムを作成してみましょう。このプログラムは一見難しく感じられると思いますが、LED の命令の特徴を理解することで簡単に実現できます。

先ほど設定エリアの説明で、「**点ける（ON）**」及び「**消す（OFF）**」は、**それぞれ LED を「点けっぱなし」「消しっぱなし」で次の命令に進む**と解説しました。これをうまく利用することで、両方の LED を点けたり、LED を左右独立して光らせたりすることができます。それでは、まず「**プログラムを開始したら右 LED を点けっぱなしにして、プログラムを終了するときに右 LED を消す**」という部分をプログラミングしてみましょう。

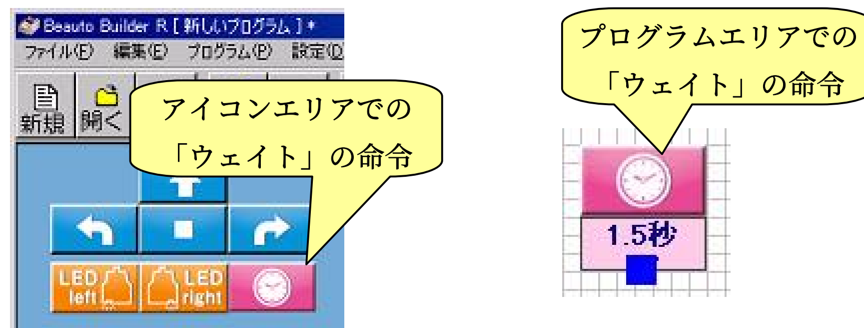


ちなみに、このプログラムを実行すると、右の LED が点けっぱなしになります（矢印がどこにもつながっていない命令を実行すると、「終了」と同じくプログラムが終了します）。

続いて、「左の LED を 1 秒間隔で点滅させる」部分をプログラミングしましょう。この「1 秒間隔で点滅」という部分をもう少し細かく分解すると、**「1 秒間光らせる」と「1 秒間消す」の二つの命令のくり返し**になります。「1 秒間光らせる」という命令は、先ほど作成したプログラムのように「1.0 秒間点ける」という命令で実現できます。それでは、下図のように左 LED の命令をプログラムエリアに二つ追加し、両方とも「1 秒間点ける」に設定しましょう。

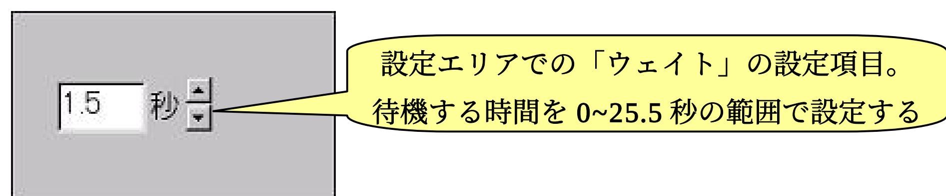


では、「1 秒間消す」命令はどうすれば実現できるでしょうか？LED の命令に「??秒間消す」という設定があればすぐに解決するのですが、あいにくこのような設定は存在しません。そこで、新しい命令「ウエイト」を使います。ウエイトの命令は、アイコンエリア及びプログラムエリアでは下記のように表示されます。

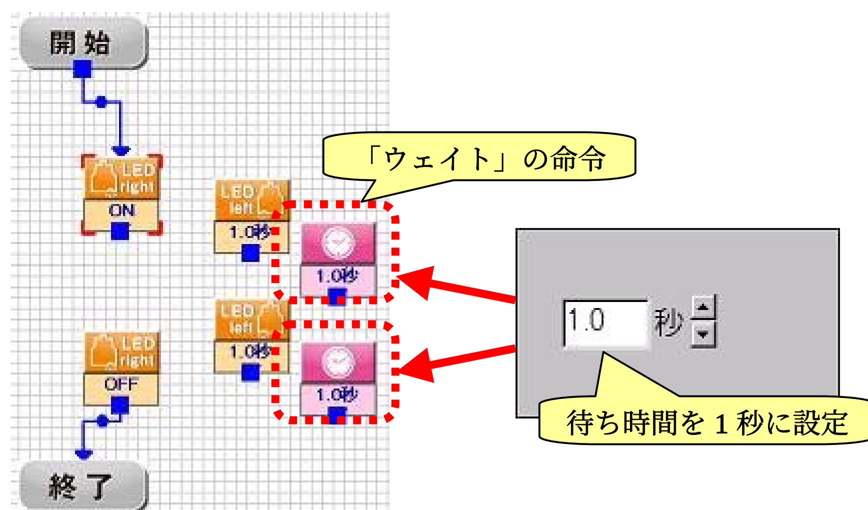


ウエイトは、「ロボットを現在の状態に保ったまま次の命令まで待つ」という命令です。例えば、ロボットのモーターが回っていたり LED が点いている状態でウエイトを実行すると、モーターや LED が動作した状態で指定した時間待ちます。ということは、「右の LED が ON、左の LED が OFF」という状態で、1 秒間のウエイトを実行することで、先ほどの「1 秒間消す」という状態を作り出すことができます。

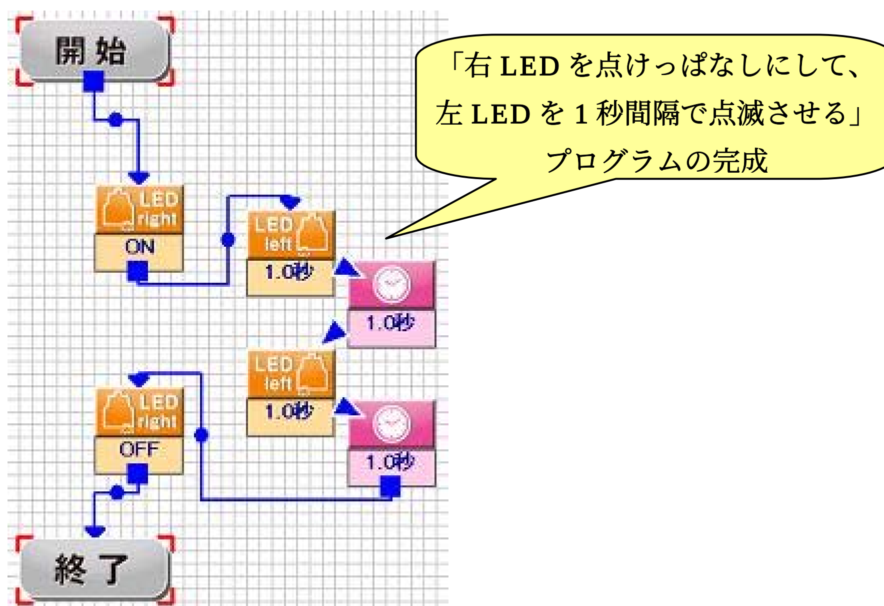
ウエイトの命令の設定項目は、待ち時間のみです。設定エリアでは下図のように表示され、待つ時間の長さを 0~25.5 秒の範囲で設定します。数値の設定方法は他の命令と同じです。



それでは、ウエイトの命令をプログラムエリアに二つ追加し、両方とも待ち時間を 1.0 秒に設定しましょう。



これで、プログラムに必要な命令が全て揃ったので、アクションブロックの矢印をつないでいきます。下図のように矢印を接続し、プログラムが完成したらロボットに書き込んで実行してみましょう。



ここまでの説明で、モーターや LED を自由に動かすプログラミングができるようになったと思います。しかし、まだここまでの解説では「開始」から「終了」まで一本道のプログラムしか作ることができません。次ページからは、センサを使用してロボットに条件分岐をさせるプログラミングを作っていきます。

分岐のプログラムに進む前に、ここまでの内容の復習として、次の課題に調整してみましょう。

プログラム課題

- 1 秒間隔で右 LED を 2 回、左 LED を 3 回点滅させるプログラムを作成しましょう
- １辺の長さが 30cm の正方形を紙に書き、それを一周なぞるプログラムを作成しましょう

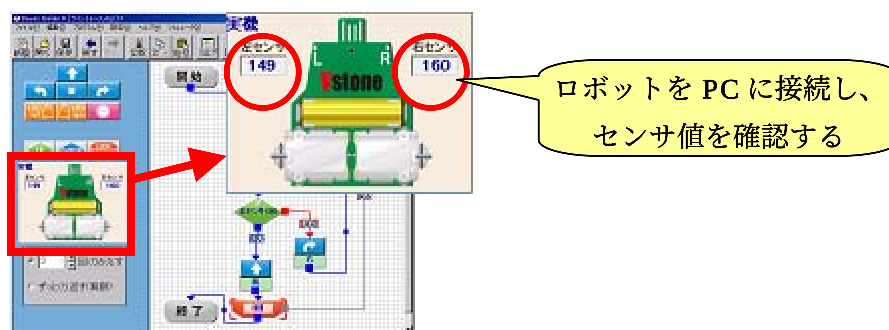
（解答例は、本書最後の「課題の答え」に掲載しています）

3.センサを使うプログラムの作成

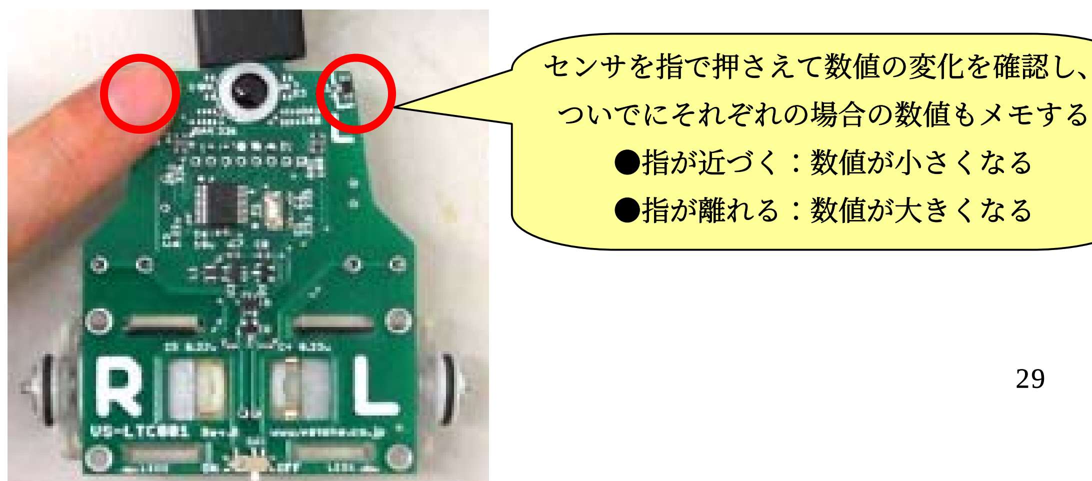
最初に説明したとおり、ロボットには前方裏側に赤外線センサが二つ付いています。この赤外線センサによって地面の色の濃さや明るさなどを認識することができ、「白い地面にひいた黒い線を辿って動く」というプログラミングもできます。このような動作を「ライントレース」と呼び、ロボットプログラミング学習の課題やロボットの競技会などでも広く取り入れられています。本章ではロボットでライントレースをするプログラムの作成を目標にしたいと思います。

3-1.センサの反応の確認

プログラミングを始める前に、ロボットのセンサの動きについて確認してみましょう。まず PC にロボットを接続して、画面左の「センサエリア」を見てください。USB 延長ケーブルを持っている場合は、ケーブルを使ってロボットを接続してください。一番初めのインストール・動作確認の際にも簡単に説明しましたが、センサエリアには現在のロボットのセンサ値が表示されます。



それでは、ロボットを USB 延長ケーブルで接続している場合は、ロボットを持ち上げてセンサを指で押さえたり指を離したりしてみましょう。このときセンサエリアの数値にも注目し、それぞれの場合で数値がどのように変化するか観察してみましょう。センサ値は、センサを指で隠すと数値が小さく、指を離すと大きく変化するはずです。このとき、それぞれの状態でどれくらいの数値になるかもメモしておいてください。

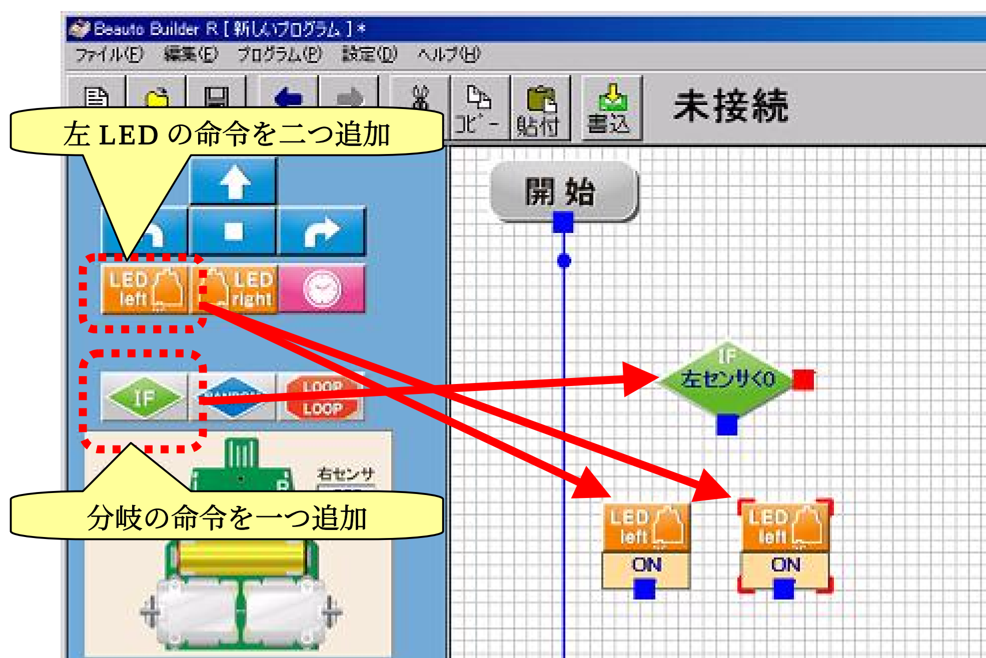


3-2.分岐の使い方

それでは分岐を使ったプログラミングをします。まずは課題として、「左センサを指で押さえたら左 LED が光り、離すと消える」というプログラミングを作成してみましょう。

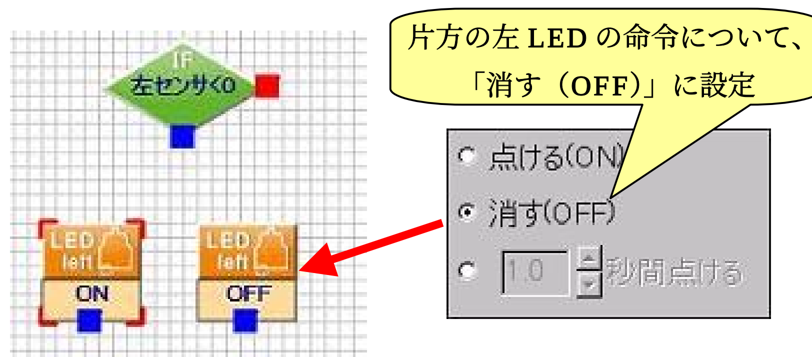
まず、センサを使って分岐を行なう命令を使ってみましょう。現在プログラムを作成中の場合は、メニューより「ファイル」→「新規作成」を、もしくはツールバーのボタンをクリックして、新しいプログラムの作成を開始してください。

続いて、下図のように「左 LED」の命令を二つ、「分岐」の命令を一つプログラムエリアに追加してください。分岐の命令は緑色のひし形のブロックです。

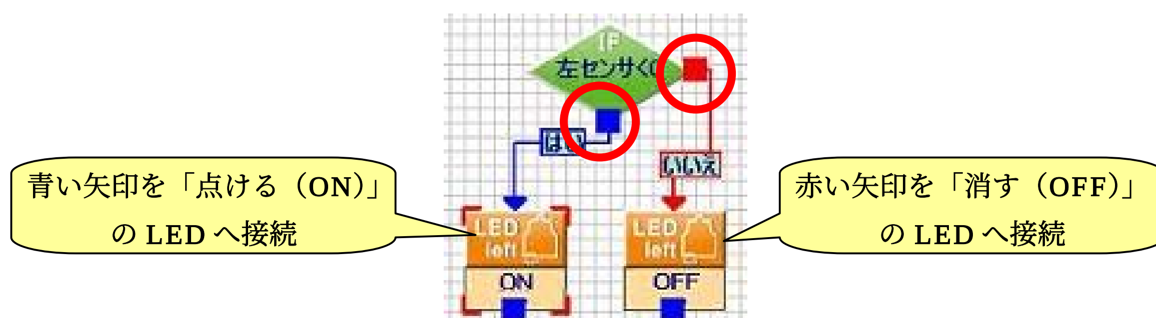


分岐の命令の設定については、長くなるので次に説明します。先に他の命令の設定と矢印の接続を行ないます。

プログラムエリアに追加した左 LED の命令は、最初は二つとも「点ける（ON)」の設定になっています。今回のプログラムでは、センサの反応によって LED を点けたり消したりするので、二つの左 LED のうち一つを「消す（OFF)」に設定してください。

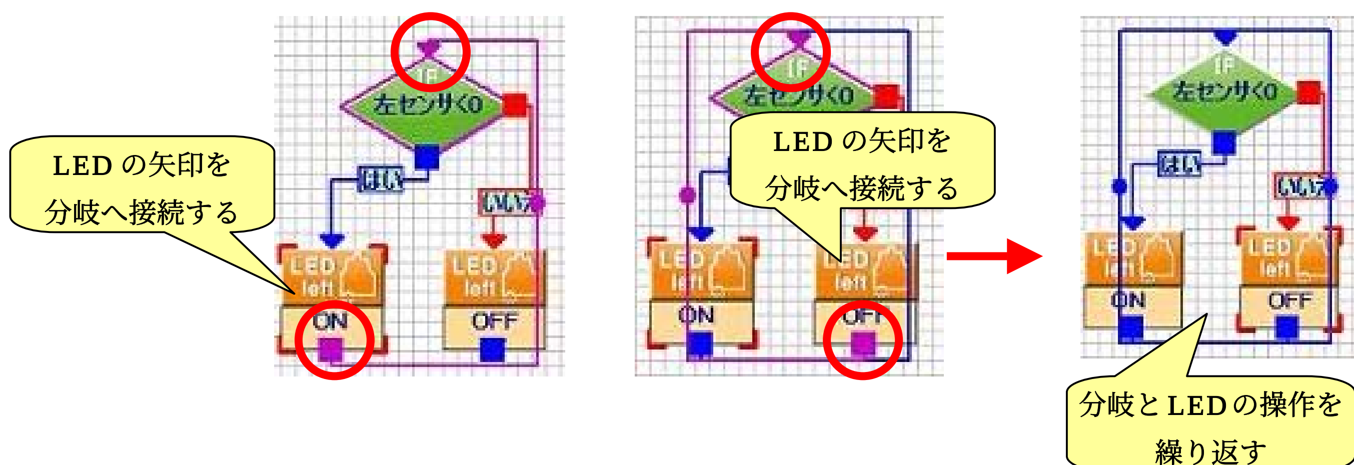


次に分岐の矢印を左 LED の命令に接続します。これまで登場したアクションブロックは全て矢印が一つだけ備わっていましたが、分岐の矢印は二つ矢印が備わっています。プログラムを実行した時には、決められた条件が成立するかにより、二つの矢印のどちらに進むかをロボットが判断します。それでは、下図のように青い四角の矢印を「点ける（ON)」の LED へ、赤い四角の矢印を「消す（OFF)」の LED へ、それぞれ接続してください。

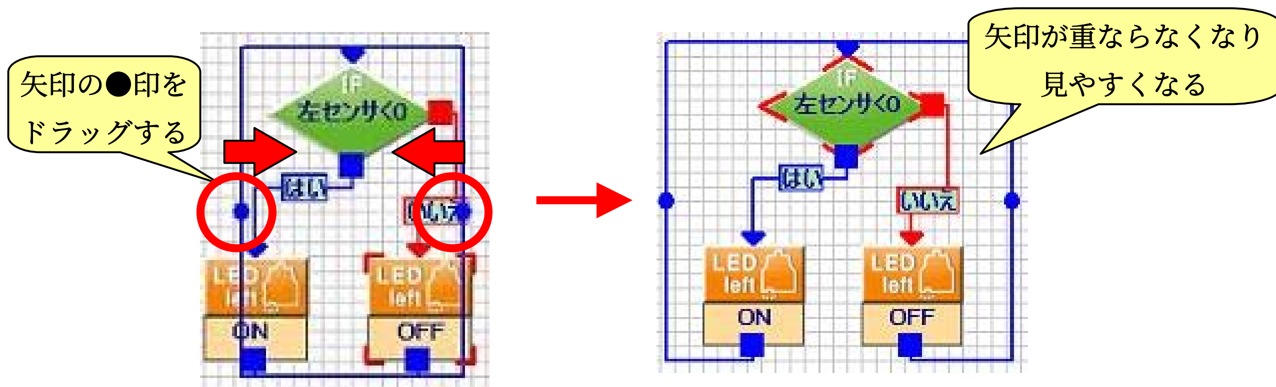


続いて、左 LED の矢印を接続します。これまでのプログラムでは、このような場合「終了」に矢印を接続していましたが、今回のプログラムは、**センサの状態が切り替わるごとに LED の点灯も切り替える**必要があります。そのため、**一度分岐からどちらかの命令へ進んでも、再び分岐に戻ってセンサの状態を確認しなおす**必要があります。

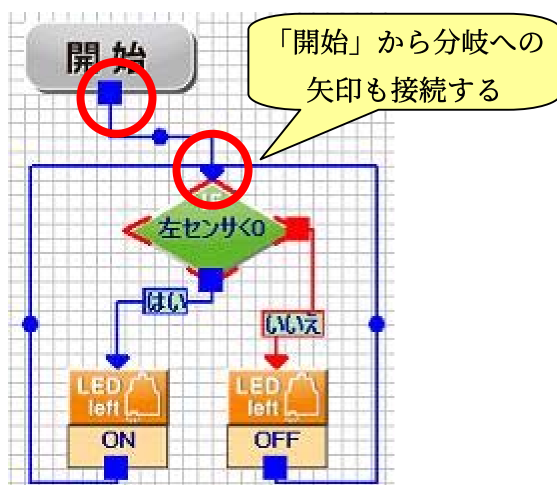
それでは下図のように、LED の命令の矢印を分岐の命令に接続してください。



ちなみに、矢印を接続すると、お互いの矢印が重なり合ってプログラムが大変見づらくなります。そこで、**矢印の途中にある●印をマウスでドラッグ**してください。ドラッグすると矢印の通り道を動かすことができます。これで右下図のように矢印が重ならないようにすると、非常にプログラムが見やすくなります。

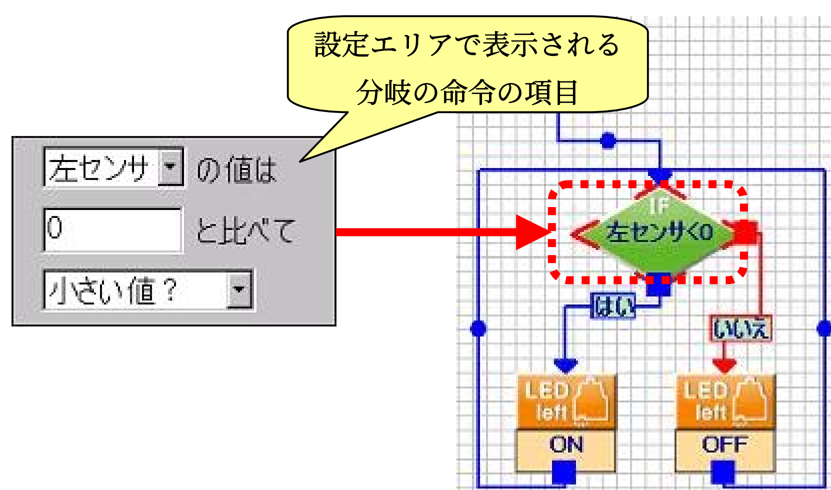


また、「開始」から分岐への矢印も接続しましょう。これで矢印の接続は完了です。続いて分岐の条件の設定を行ないます。

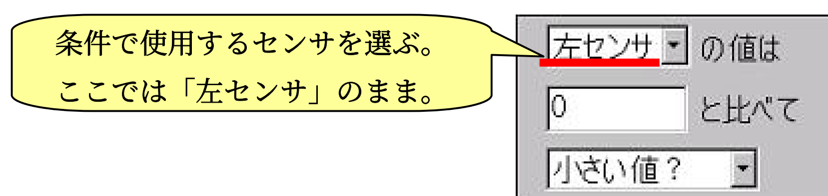


現在のプログラムでは、**最も肝心な「センサがどのようなときに LED を点けるのか、また、どのようなときに消すのか」という条件が設定されていません**。この部分を設定して、プログラムが完成します。それではその設定について、分岐の設定エリアでの設定方法と合わせて解説します。

まず、プログラムエリアから分岐の命令をクリックし、設定エリアを確認してください。設定エリアには下図のような項目が表示されます。



分岐命令の設定は 3 行に分かれており、その全てを設定する必要があります。まず **1 行目は「条件に使用するセンサ」を選択します**。今回のプログラムでは最初の「左センサ」の設定で問題ありません。逆に右センサを使用する場合は、ここで「右センサ」を選択する必要があります。



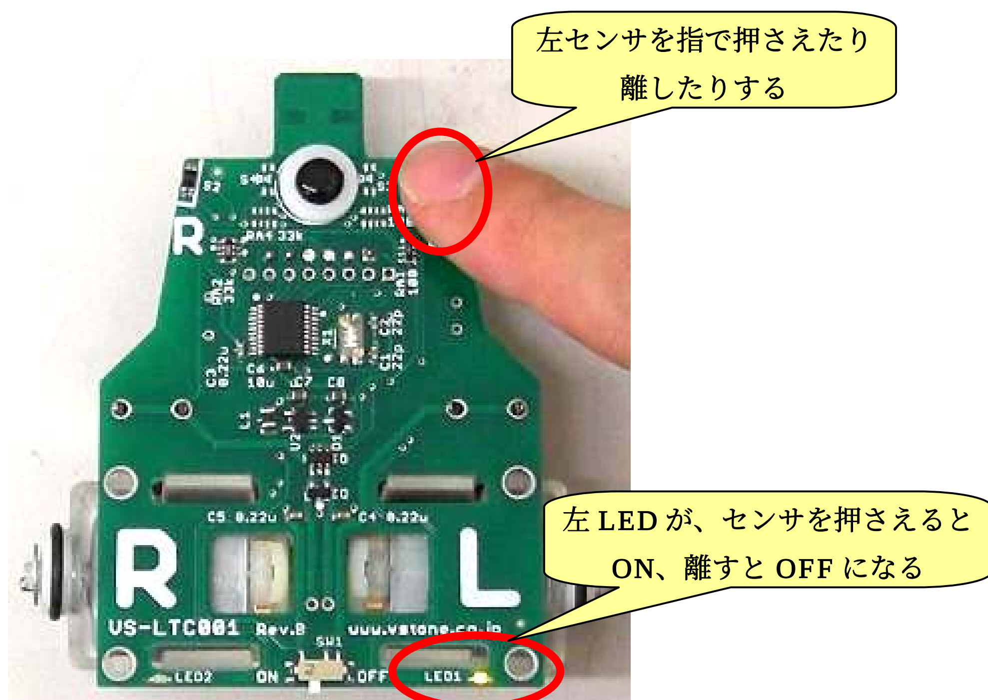
2 行目と 3 行目は、「**センサと比較する定数**」と「**条件の種類**」をそれぞれ設定します。ここはまず先に正解を書きます。下図の通り **2 行目**に「**60**」を入力、**3 行目は**「**小さい値?**」のままにしてください。

2 行目はセンサと比較する定数。
60 を入力する

左センサ の値は
60 と比べて
小さい値？

3 行目は条件の種類。
「小さい値?」のままにする

それでは、この正解が実際にちゃんと動作するのか、ロボットに書き込んでプログラムを実行してみましょう。プログラムを実行すると、ロボットの左センサを指で押さえると左 LED が点き、指を離すと LED が消えます。



それでは、このプログラムがロボットの中でどのように実行されているのか、順番を追って確認してみましょう。先ほどロボットのセンサを指で押さえて数値を確認しましたが、**センサの数値は、指でセンサを押さえると非常に小さく、指を離すと数値が大きくなった**と思います。ここでセンサの数値が「センサを指で押さえたときは 0 付近」「指を離したときには 160 付近」と仮定して、それぞれの場合におけるプログラムの順序を見てみたいと思います。

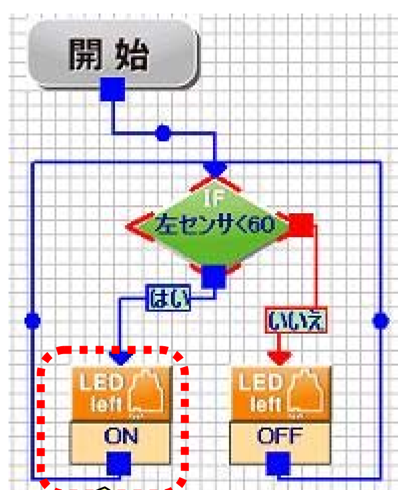
まず**センサの値が 0 付近の場合、「0（センサ値）<60」が成り立つので「はい」の矢印に進みます。**「はい」の矢印の先には左 LED を点ける命令があるので、「センサを指で押さえると LED が光る」というプログラムになります。

逆に**センサの値が 160 付近の場合、「160（センサ値）<60」が成り立たないので「いいえ」の矢印に進みます。**「いいえ」の矢印の先には左 LED を消す命令があるので、「センサから指を離すと LED が消える」というプログラムになります。

また、**左 LED を ON/OFF した後に、矢印は再び分岐につながっているので、再びセンサの情報を見てどちらかに進み、**また分岐に戻って・・・というようにプログラムをずっと繰り返します。

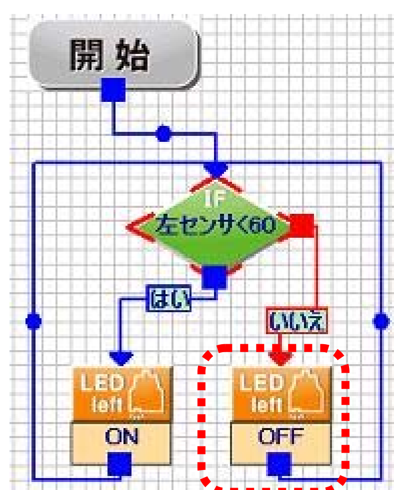
条件が「60」と比べて「小さい値？」の場合

指で押さえる＝
センサの値が 0 である



条件が成立するので「はい」に進む。
左 LED は「点ける（ON）」

指を離す＝
センサの値が 160 である



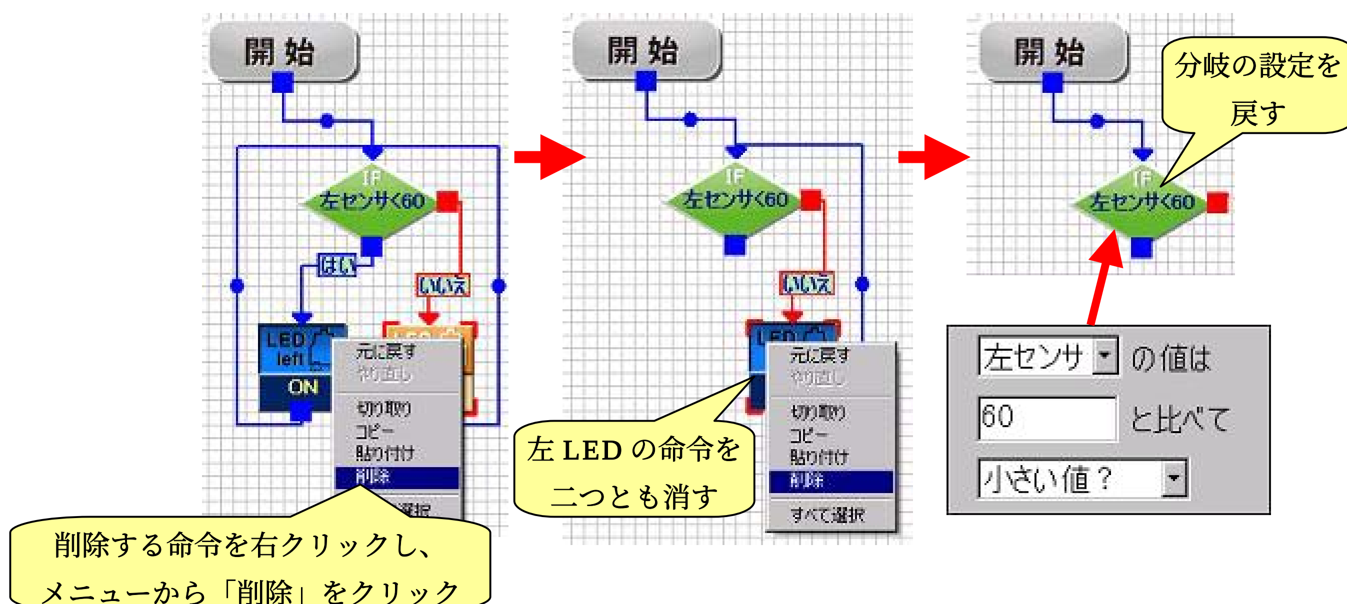
条件が成立しないので「いいえ」に進む。
左 LED は「消す（OFF）」

プログラムの動作を確認して仕組みを把握したら、次の場合にどのような動きになるのか確認してみましょう。

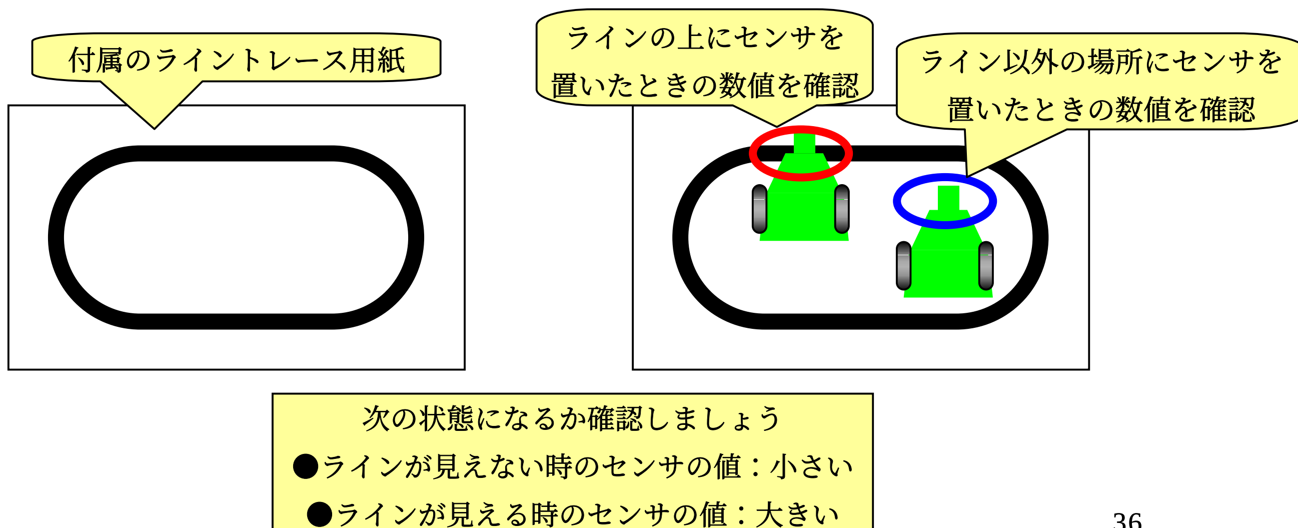
● 分岐の条件を「大きい値?」に変更したらどうなるか確認してみましょう
● 「はい」と「いいえ」の矢印を入れ替えてつないだらどうなるか確認してみましょう

3-3.ライントレースのプログラミング

それでは、ここまでで作成したプログラムを応用してライントレースのプログラムを作って見ましょう。まず、左 LED の命令を二つとも消しましょう。**命令を消す場合は、マウスカーソルを削除する命令に合わせて右クリック**し、表示される**メニューから「削除」を**クリックしてください。また、削除する命令をクリックしてキーボードの Delete キーを押しても消すことができます。また、**分岐の命令の設定を「左センサが 60 と比べて小さい値?」に戻して**おいてください。



また、商品に入っている下図のようなライントレース用のコースを準備してください。コースを用意したら、USB 延長ケーブルを持っている場合は、最初にセンサの値を確認したように「**ラインの真上にロボットのセンサを置いたとき**」と「**ラインがない場所にロボットのセンサを置いたとき」のそれぞれのセンサ値を確認**しておきましょう。

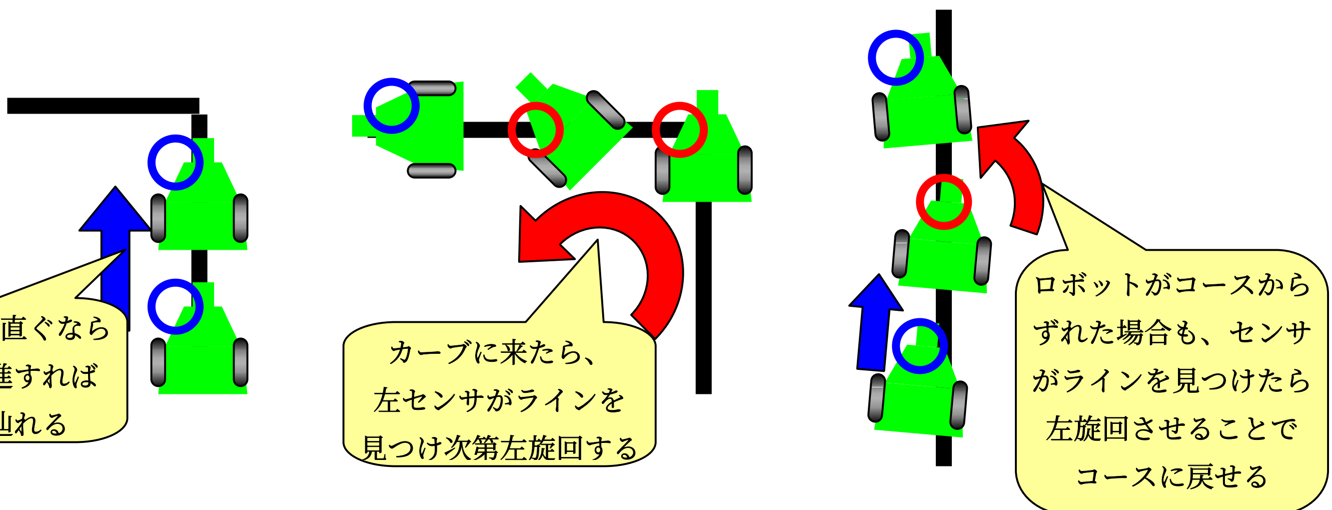


では、削除した左 LED の命令に変わってモーターの命令を追加したいと思います。しかし、モーターを動かす命令は 4 種類備わっていますが、今回のプログラムではどれを選択すればよいでしょうか。ライントレースの原理を説明するので考えて見ましょう。

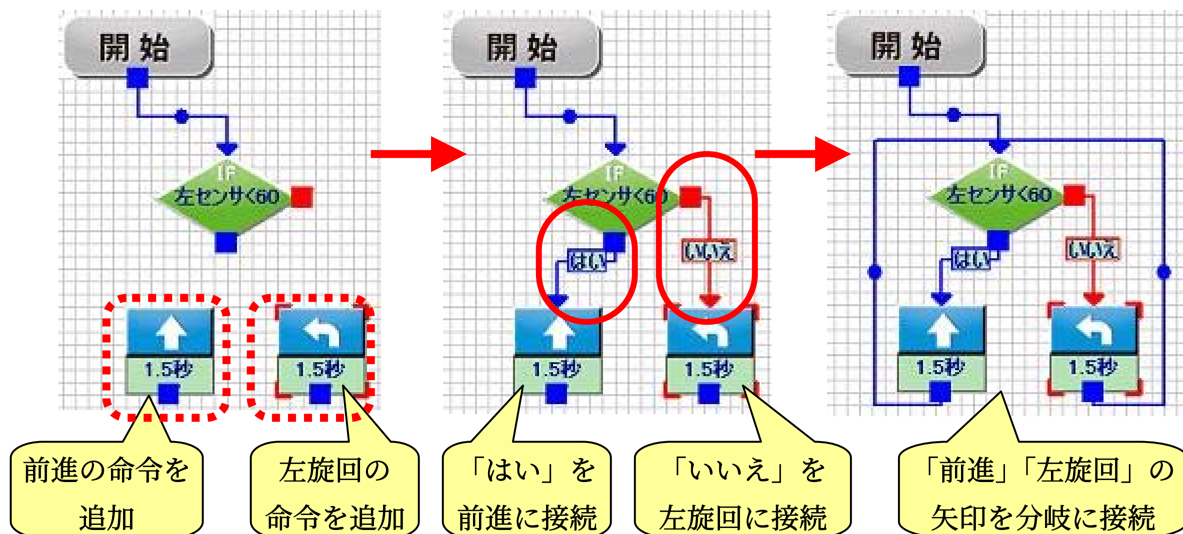
まず、ライントレースは「**ロボットが線をまたいだ状態からスタートする**」ことを前提とします。この状態では「**左右どちらのセンサもラインを見ていない**」状態になります。ここで、下図左のように**ラインが真っ直ぐであれば、そのまま真っ直ぐ進むことで自然にラインを辿って移動できる**ことが分ります。

では、下図中のように**コースの曲がり角**に来た場合はどうすればよいでしょうか。このまま真っ直ぐ行くとラインから外れてしまいます。しかし、**カーブを通過する途中で、必ず左センサがラインに反応**します。そこで「**左センサがラインを見つけたら左旋回する**」というプログラムを作成すれば、線を外れずに動くことができます。

また、下図右のように、**スタート時にロボットが少し曲がってしまった場合**も、このまま真っ直ぐ行くと線を外れてしまうので「**左センサがラインを見つけ次第左旋回する**」**ようにすれば、再びラインに戻ってくることができます**。



それでは、この説明の通り「前進」と「左旋回」の命令をプログラムエリアに追加してください。追加したら下図の通りに接続しましょう。

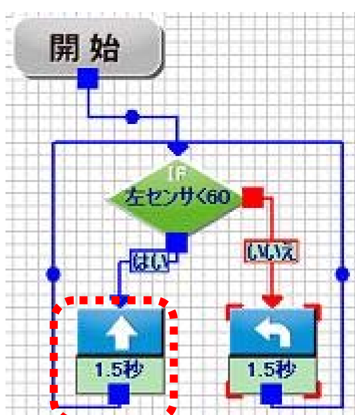


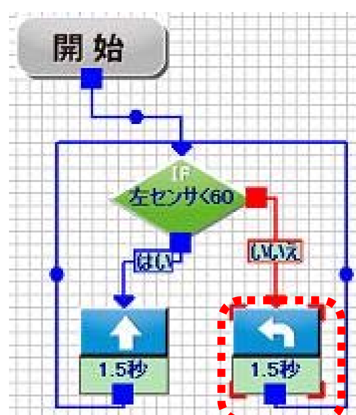
それでは、ここまでのプログラムが正しいかどうか、実際に動かす前に確認してみたいと思います。先ほどのプログラムのように**「ラインが見える場合」と「ラインが見えない場合」で、それぞれどのように動作するか**確認すると、問題が無いようです。それでは、ロボットにプログラムを書き込んで実際に動かしてみましょう。なお、コースを走らせる場合は「反時計回り」でスタートしてください。

条件が「60」と比べて「小さい値？」の場合

ラインが見えない＝
センサの値が 0 である

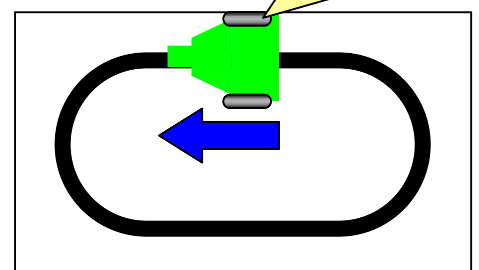
ラインが見えている＝
センサの値が 100 である





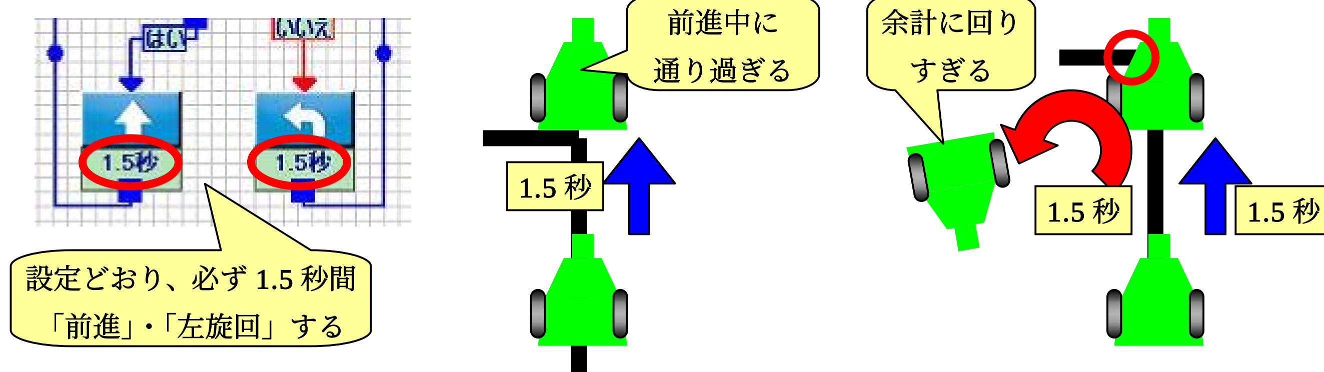
条件が成立しないので
「いいえ」に進む。
モーターは「左旋回」

反時計回りでロボットを
動作させる



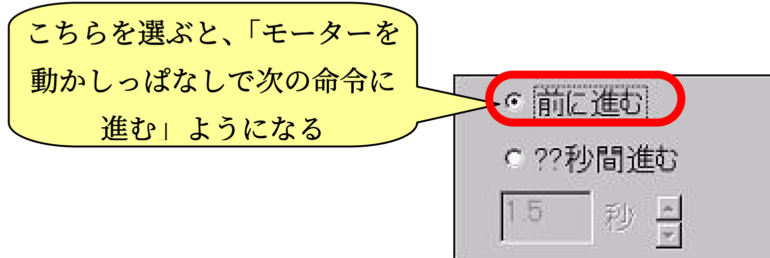
プログラムを実行すると、なぜかロボットがラインから外れたり、または極端に旋回したりして、うまくライントレースできません。プログラムの流れは正しいのになぜ正しく動作しないのでしょうか。これは、モーターの命令の設定に問題があります。

これまで説明したモーターの命令は「設定エリアで指定した秒数だけ動作する」というものでした。そこで、作成したプログラムを見直すと、**「前進」「左旋回」がいずれも「1.5 秒間動作する」**となっています。このままでは、**センサがラインを確認した後に必ず 1.5 秒モーターを動かす**ので、下図中のように**カーブを見落としたり**、下図右のように**うまくラインを見つけても 1.5 秒も余分に左旋回**したりしてしまいます。

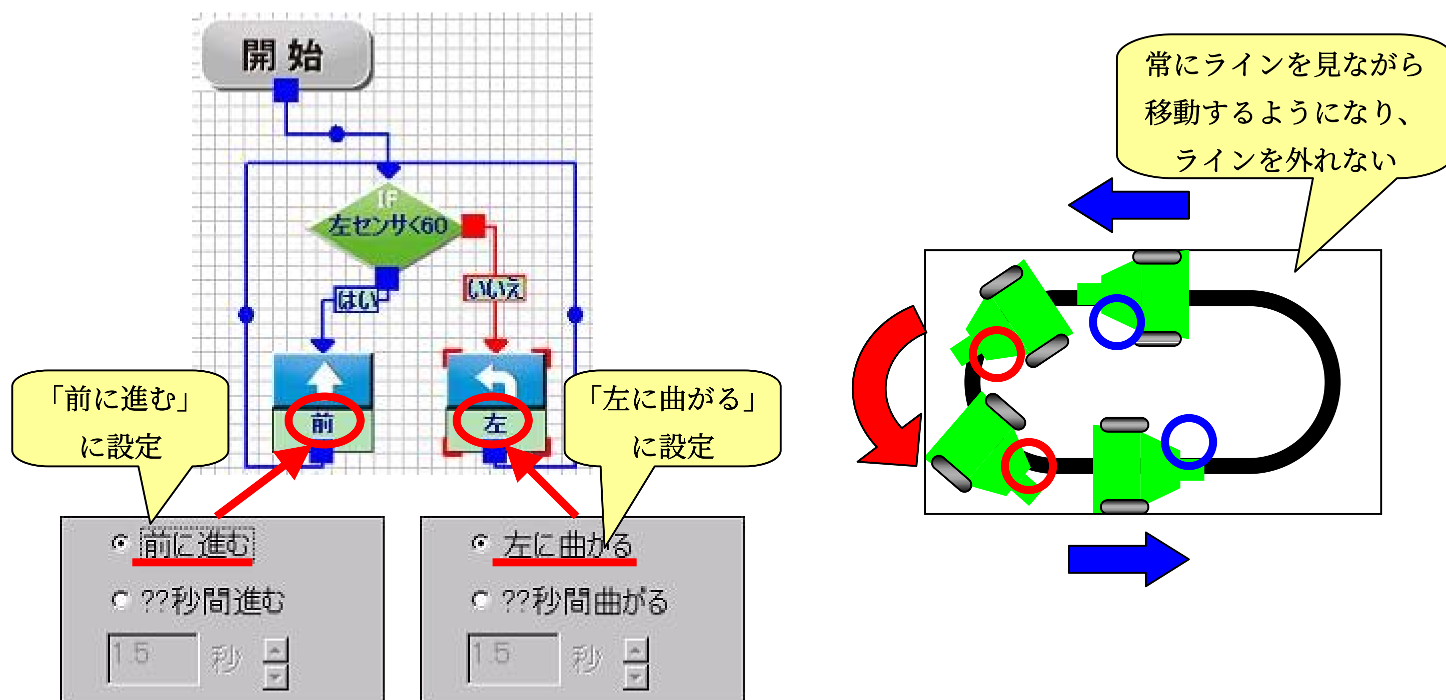


この問題を解決するには、**「1.5 秒の設定時間を短くする」**という方法が考えられますが、一番短い時間の**「0.0 秒」を設定するとロボットが動かなくなります**。これは**「0.0 秒前進・旋回する」ということは「前進・旋回しない」ということになるため**です。では「0.1 秒」に設定すると、モーターは動くようになりますが、それでも 0.1 秒間センサを見落とす時間ができてしまいます。

最も良い解決方法は、時間を指定せずに**「モーターを動かしつつセンサを確認する」**という方法です。これをどのように設定するかについては、実は似たようなことを LED の命令で既に学習しています。LED の命令では「点ける（ON）」「消す（OFF）」を選択すると、「LED を点けっぱなし」「LED を消しっぱなし」の状態で次の命令に進めることができました。これと同じように**モーターの命令にも「モーターを回しっぱなし」「モーターを止めっぱなし」で次の命令に進む設定があります**。



それでは、「前進」「左旋回」の命令を、左下図のように「前に進む」「左に曲がる」に変更してください。変更したらプログラムをロボットに書き込んで実行し、正しくライントレースできるようになるか確認してみましょう。



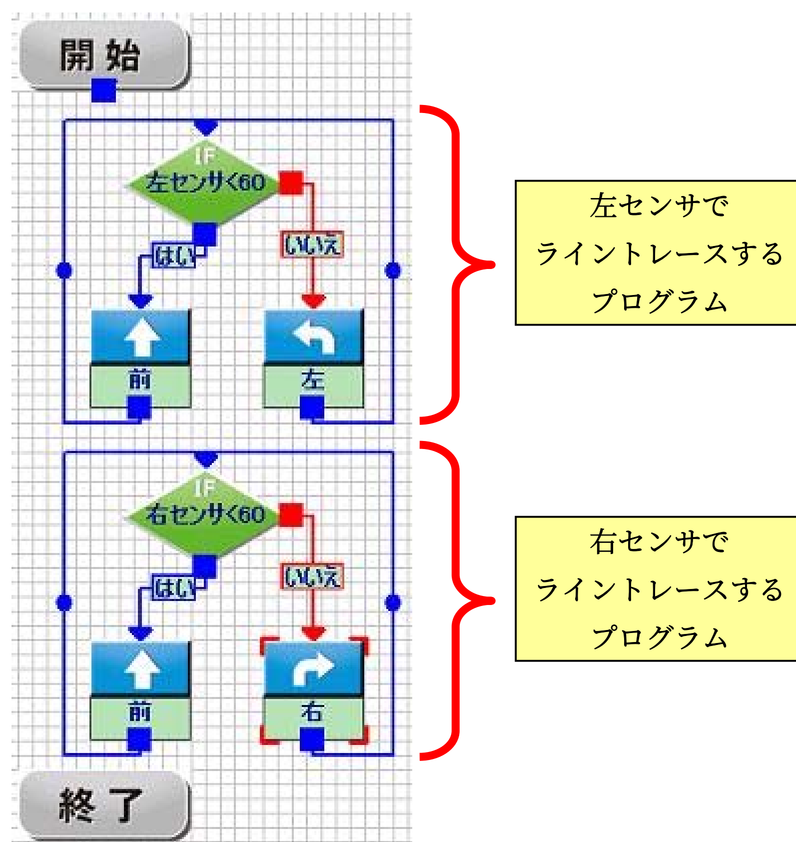
プログラムが正しく動作するようになったら、今度は**右センサを使用してライントレースするプログラム**に改造してみましょう。これは、ほとんど同じプログラムで作ることができます。ヒントは「**右センサを使うこと**」「**旋回する方向を逆にすること**」「**時計回りでコースを走らせること**」の三つです（解答例は、本書最後の「課題の答え」に掲載しています）。

プログラムを正しい設定に直してもロボットがラインから外れてしまう場合は、モーターのスピードが速すぎる可能性があります。この場合は、モータースピードの設定で両方のモータースピードを同じだけ減らして遅くしてみましょう。

3-4.左右センサを使ったライントレース

ここまでで説明したプログラムは、左センサ、もしくは右センサのどちらか一つだけを使っているため、時計回り・反時計回りのどちらか一方でしか正しく動作しませんでした。これを、左右両方のセンサを同時に使うプログラムにすれば、コースをどちらの方向に走らせてもライントレースできるようになります。それでは、両方のセンサを使ったプログラムを作成してみましょう。

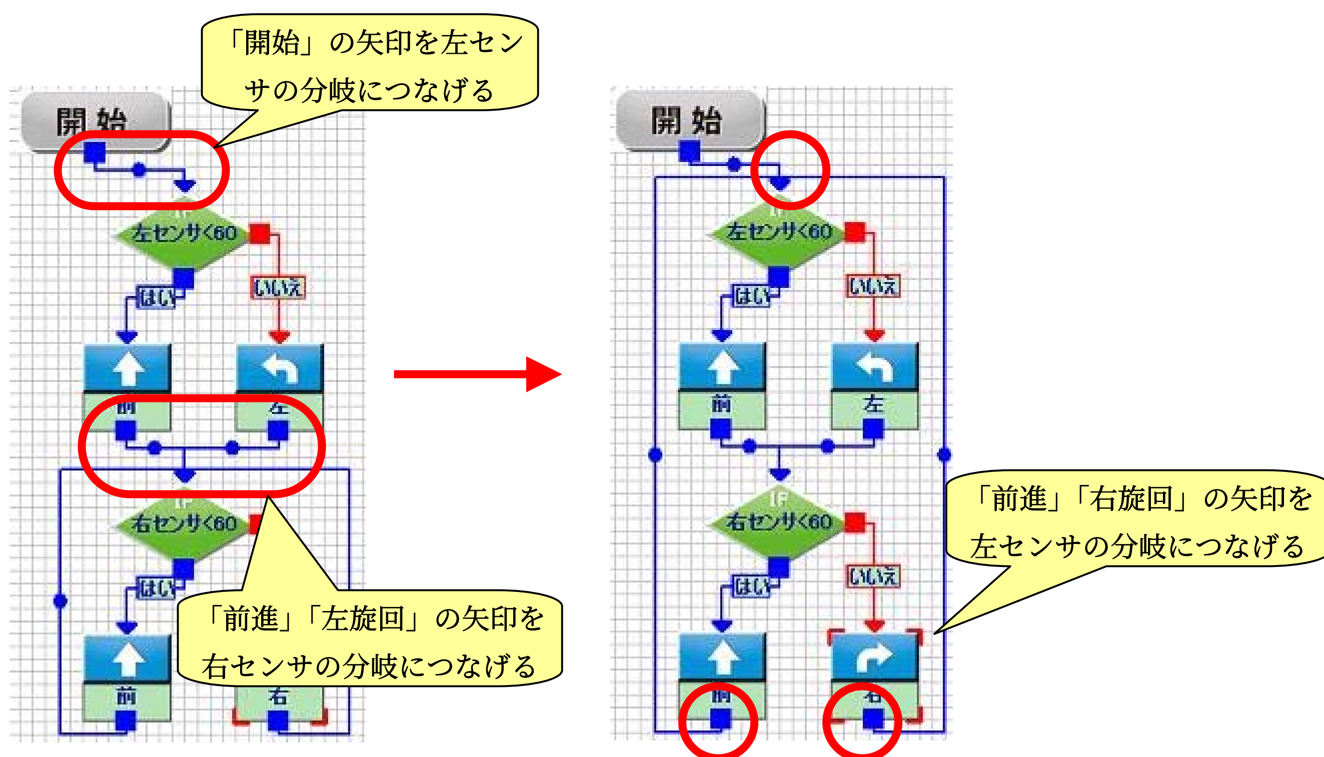
まず、左右センサでそれぞれライントレースするプログラムを作成してください。左センサでライントレースするプログラムは先ほど作成したものとまったく同じです。また、右センサでライントレースするプログラムは、分岐の「左センサ」を「右センサ」に、「左旋回」の命令を「右旋回」に、それぞれ変更してください。「前進」「左右旋回」の命令は、秒数を指定しない（「??秒間～」の方を選択しない）ようにしてください。



このプログラムを改造して、両方のセンサを見てライントレースするプログラムにします。

現在のプログラムでは、**一つのセンサ（分岐）の間で「前進」と「旋回」を行き来する構造**になっているので、まずは「**一つのセンサを見て「前進」か「旋回」に進んだ後、次のセンサ（分岐）に進む**」という**構造にしてみます**。左センサの分岐からつながっている「前進」と「左旋回」の矢印を、右センサの分岐に接続してみましょう。ついでに「開始」の矢印を左センサの分岐に接続しましょう。

矢印をつなぎかえることで、「左センサを見た後に右センサを見る」という構造にすることができました。しかし、まだ右センサの方は「**右センサと「前進」・「右旋回」を行き来する**」という構造になっているので、右センサの分岐につながっている「前進」と「右旋回」の矢印も、左センサの分岐に接続しましょう。



これで、プログラムを実行したときに左右両方のセンサを見るようになりました。それでは、ロボットにプログラムを書き込んで実行してみましょう。

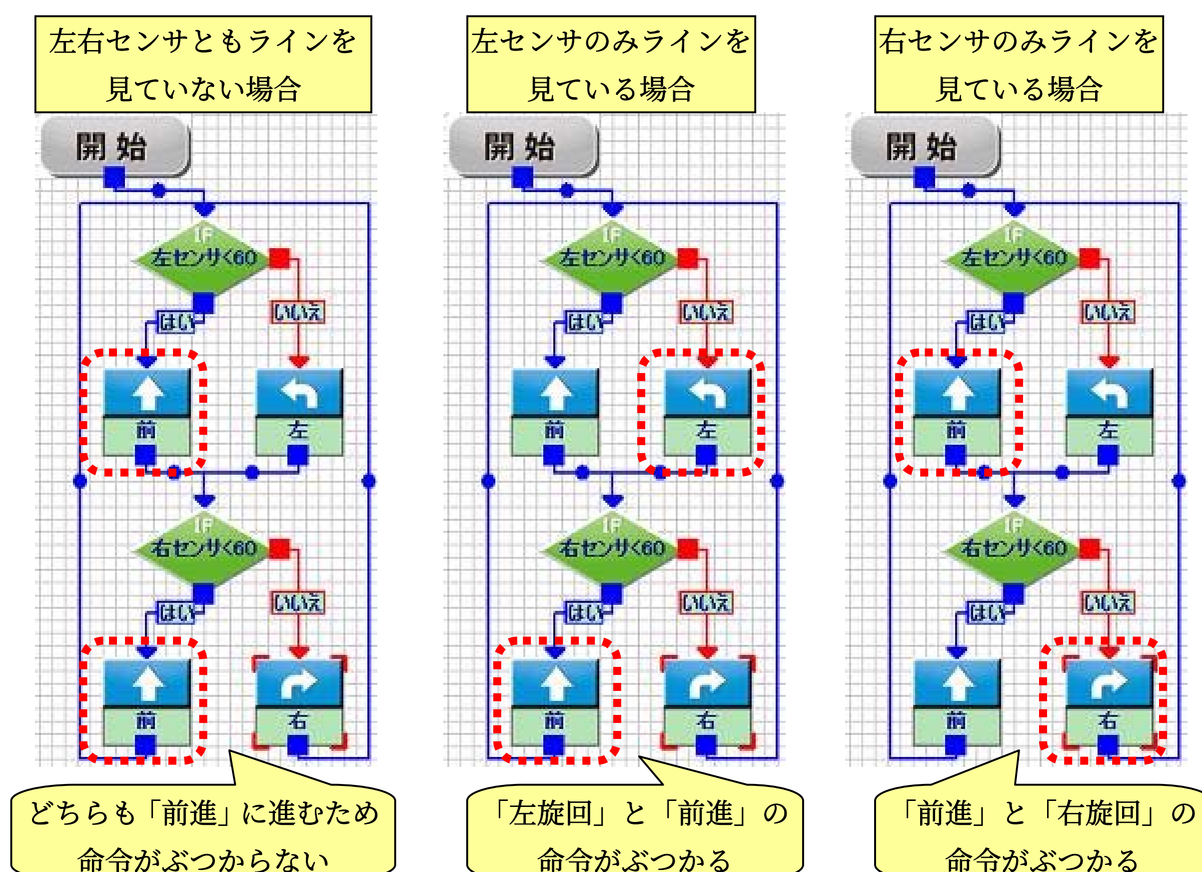
プログラムを実行すると、なぜかロボットの動きがおかしくなると思います。**ラインを見ているはずなのにまったくラインを無視して前進する**ようになってしまいます。プログラムは一見正しいように思えますが、どこかに問題が含まれているようです。

では、プログラムのどこが問題なのか、命令の実行順序を追って確認してみましょう。

まず、**左右のセンサがどちらもラインを見ていない場合**ですが、最初に登場する左センサの分岐は正しいプログラムから変更していないため、正しく「前進」に進みます。続いて登場する右センサの分岐についても、正しいプログラムから変更していないため、正しく「前進」に進みます。**どちらの分岐でも同じ「前進」の命令が選ばれるので、ロボットは正しく前進します。**

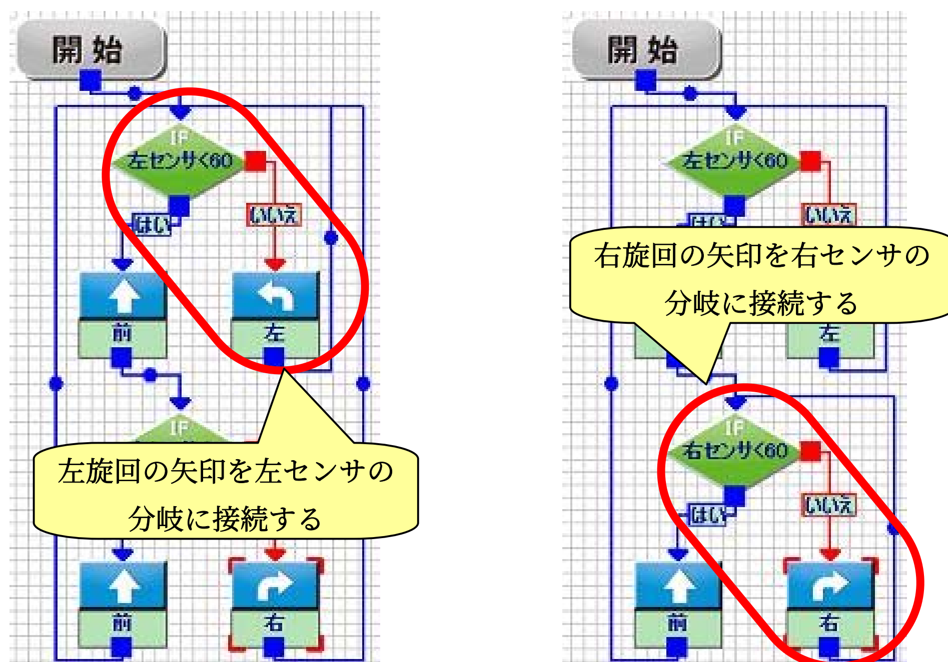
次に、**左センサがラインを見つけた場合**、左センサの分岐は「左旋回」に正しく進みます。そして続いて**右センサの分岐に進んだとき、右センサはラインを見ていないので「前進」に進みます**。このとき、**「前進」と先ほどの「左旋回」の命令がぶつかり合います**。実はこの部分が問題の原因です。**「左旋回」の命令でロボットは左モーターを止めようとしますがモーターはすぐに止まりません。そうしている間に「前進」の命令が与えられ、止めようとした左モーターをすぐ動かしてしまいます。**その結果、**ロボットは前進し続けます。**

逆に、**右センサがラインを見つけた場合**も、**右センサの分岐は「右旋回」**に正しく進みますが、**左センサはラインを見ていないので、一つ前の左センサの分岐で「前進」**に進んでおり、**「前進」と「右旋回」の命令がぶつかり合います**。ここでも、ロボットは「右旋回」の命令で右モーターを止めようとしますが、プログラムが一巡して次の左センサの分岐で「前進」がすぐに出てくるので、再び右モーターを動かし前進し続けることになります。

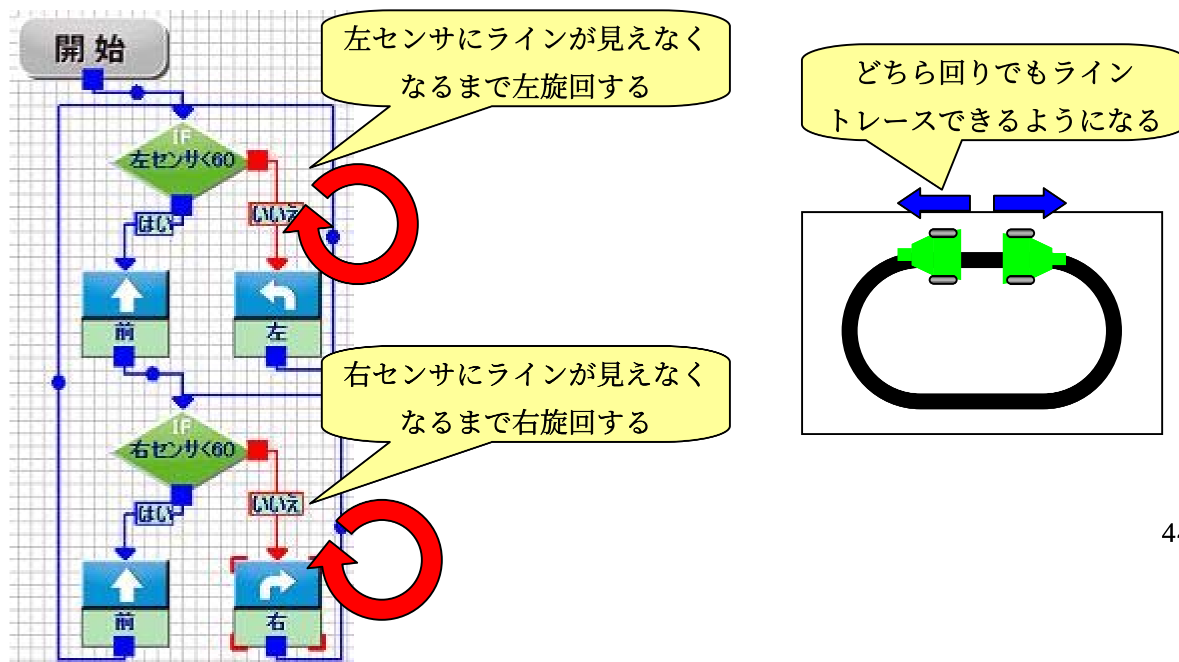


プログラムの順序を確認して「**二つの分岐の命令がぶつかるのはよくない**」ということが分りました。しかし、二つの分岐を通るようにプログラムしないと左右のセンサを見ることができません。どうすれば「**二つの命令がぶつからない**」、且つ「**両方のセンサを見る」プログラム**ができるでしょうか。答えは「**片方のセンサの処理＝旋回命令を終えてから次のセンサに進む**」という方法です。

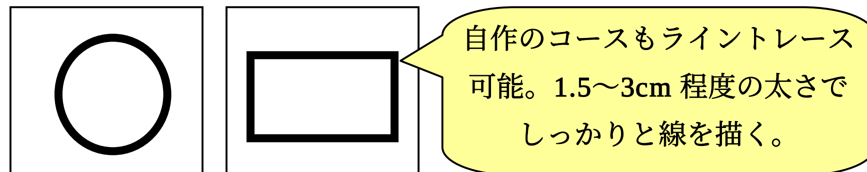
それでは、現在のプログラムの矢印を下図のようにつなぎ変えてみましょう。左旋回は左センサの分岐に、右旋回は右センサの分岐に、それぞれ矢印をつなぎ変えます。つなぎ変えたらプログラムを実行してみましょう。



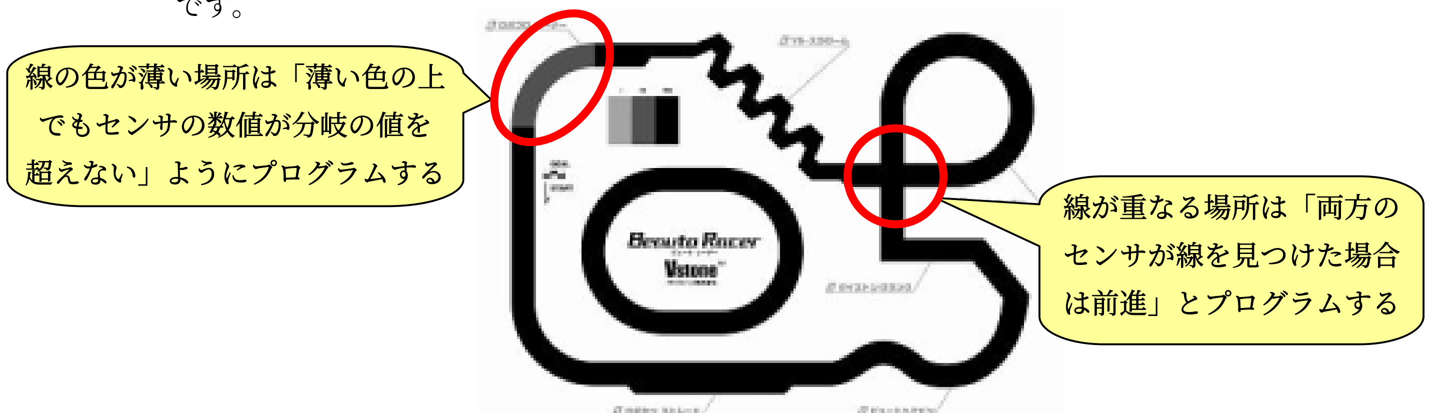
このプログラムを実行すると、左右のセンサがラインを見つけたら、「ラインを見つけたセンサがラインを見えなくなるまで旋回する」ようになります。センサがラインを見ていない状態であれば次のセンサに進みます。これで、「旋回」と「前進」の命令がぶつかり合うことがなくなり、正しくライントレースできるようになります。



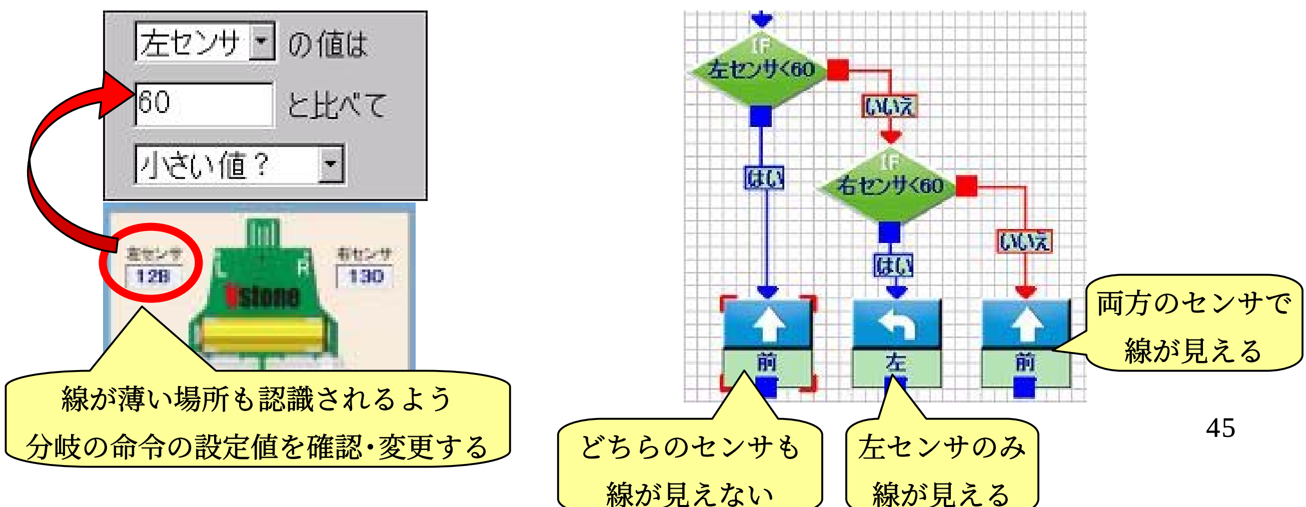
ここまで解説したプログラムなら、下図のように単純な図形のコースであればライントレースできるようになります。白い紙に黒いマジックなどで線を引いたり、PC 上で描いたコースを印刷したりしてロボットに走らせて見ましょう。注意として、線はなるべく太く（1.5～3cm 程度)、また濃くはっきりと描きましょう。また、モータースピードが速すぎると、勢いによってロボットがコースから外れやすくなるので、その場合はモータースピードを遅めに調整しましょう。



また、付属 CD に収録されている上級者向けのライントレースコースも、このプログラムを基本として走ることができますが、一部の場所はもう少し複雑なプログラミングが必要になります。特にポイントとなるのは「線の色が薄い場所」と「線が重なっている場所」です。



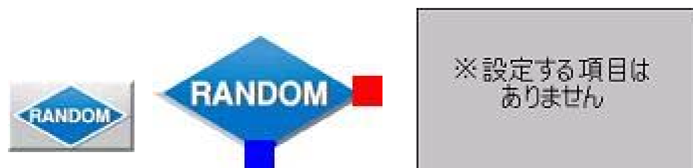
「線の色が薄い場所」は、その場所でのセンサの数値を調べ、分岐命令でそこも線として認識されるように設定値を書き換えます。「線が重なる場所」は、「両方のセンサが線を見つけた場合」のプログラミングを行います。この場合「前に進む」とラインを外れないので、そのようにプログラミングしましょう。



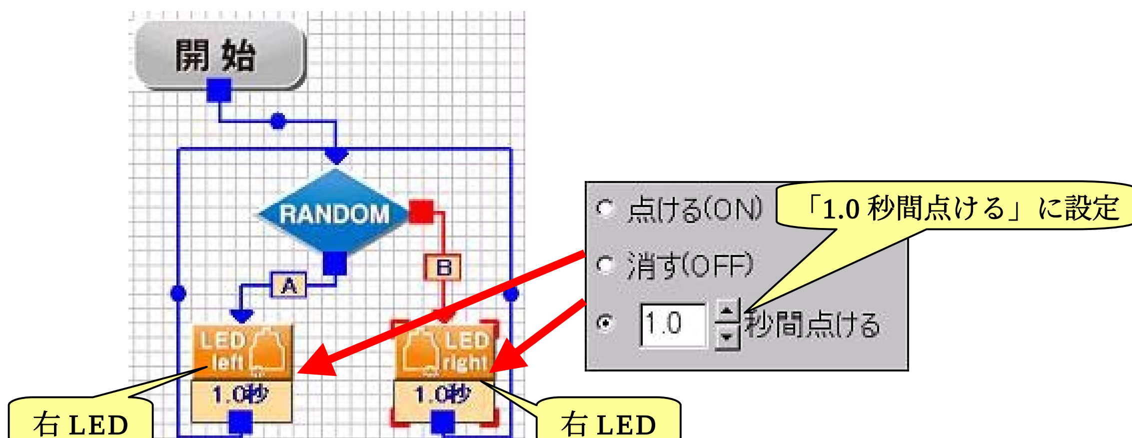
3-5.ランダムの命令

プログラムを分岐させる命令には、もう一つ「ランダム」があります。ランダムは、センサを使った分岐のように、分岐させるための決まった条件を設定しません。分岐のどちらに進むかは、プログラム実行中にロボットが勝手に決めます。ランダムの命令の使い道としては「ルーレット」のように偶然の要素が必要な場合、「探査」「相撲」などのように動きに意図しない多様性を組み込みたい場合などが挙げられます。

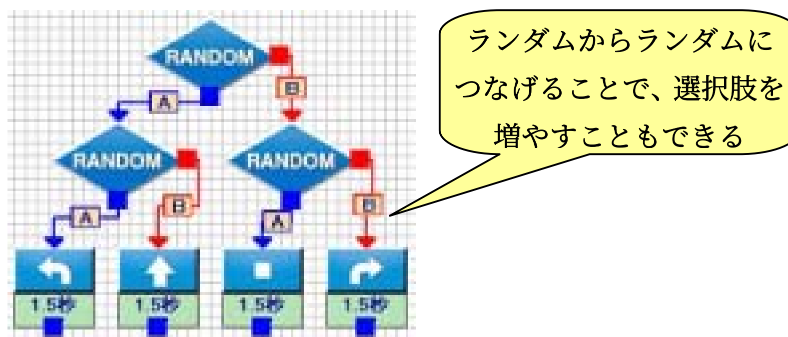
ランダムの命令は、アイコンエリア・プログラムエリアでそれぞれ下図のように表示されます。また、条件を設定しないため、設定エリアでは何も設定項目がありません。



それでは、ランダムを使ったプログラムのサンプルとして「左右の LED をランダムで 1 秒ずつ光らせてみる」というプログラムを作成してみましょう。ランダムの命令、及び左右 LED の命令を一つずつ追加し、下図のように矢印を接続してください。また、LED の命令はそれぞれ「1.0 秒間点ける」に設定してください。プログラムを作成したらロボットに書き込んで実行してみましょう。



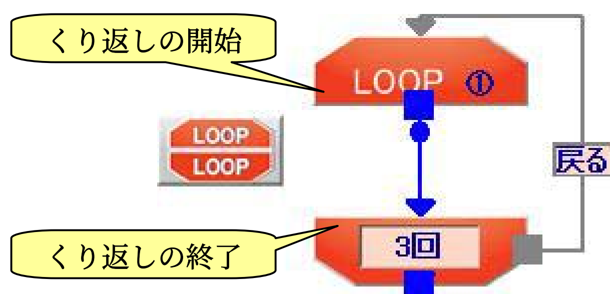
また、ランダムの命令は基本的に「A」「B」の二択ですが、複数つなげることで選択肢を増やすことができます。例えば下図のように、一つのランダムの先に更に一つずつランダムを接続すれば、結果的に四択のプログラムを作ることもできます。



4.くり返しを使うプログラムの作成

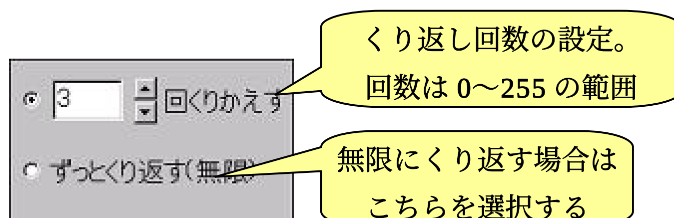
例えば「**1 秒間隔で LED を 100 回点滅させるプログラム**」を作成する場合、これまで説明してきた方法では、「**LED を 1 秒間点ける**」と「**1 秒間のウエイト**」の二つの命令を、交互に **100 個ずつ並べなければいけませんでした**。しかし、この方法だとプログラミングに時間がかかりとても大変です。また、ロボット本体もこれほど長いプログラムを記録することができません。そこで、**同じ命令を何度も実行する場合は「くり返し」の命令を使用します**。

くり返しの命令は、アイコンエリアとプログラムエリアでは下図のように表示されます。

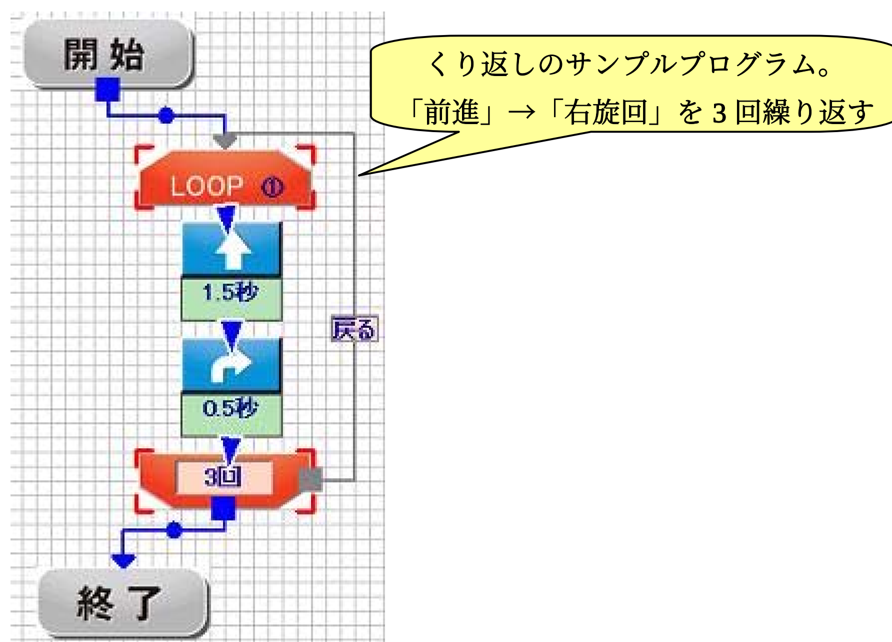


くり返しの命令は、**一度に二つのアクションブロックが追加されます**。これはそれぞれ「**くり返しの開始**」と「**くり返しの終了**」を表しています。プログラムが「開始」から「終了」までを実行するように、**くり返しにも開始と終了があり、この間にはさまれた命令が、設定した回数だけ繰り返されます**。また、くり返しの開始と終了も、矢印を正しくつなげないとくり返しが正しく行われません。

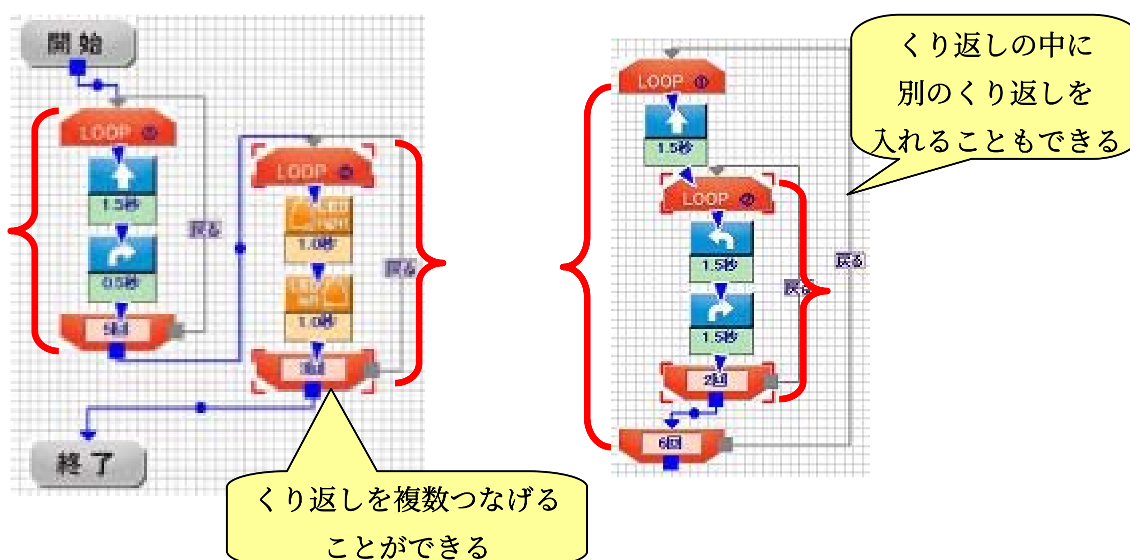
設定エリアでのくり返しの表示は下図のようになります。くり返しで設定できる項目は**くり返す回数**で、**0~255 回の範囲**で設定できます。ちなみに、**くり返し回数を 0 に設定した場合、くり返しの中身を一度も実行せずにプログラムを進めます**。また、ずっとプログラムをくり返す場合は「ずっとくり返す（無限）」を選択します。



それでは早速くり返しを使ったプログラムを作成してみます。下図のプログラムを作成して実行してみましょう。プログラムを実行すると、**ロボットが「1.5 秒前進」→「0.5 秒右旋回」という動作を 3 回くり返してプログラムを終了します。**また、設定エリアでくり返し回数を変更すると、動作をくり返す回数が 3 回から変化します。

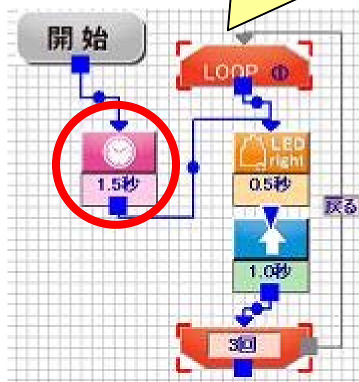


くり返しは、一つのプログラムで最大 7 個まで使うことができます。くり返しを複数組み合わせることで、より複雑なくり返しもプログラミングできます。しかし、矢印のつなげ方をしっかり把握していないと、くり返しが正確に行なわれないので、注意が必要です。

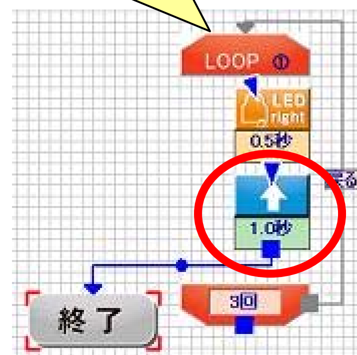


くり返しを次のように接続すると、プログラムを書き込むときに警告が表示されます。

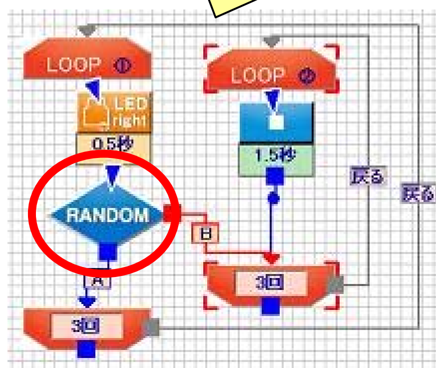
×くり返しの開始を通らず
いきなりくり返しの中身に
矢印を接続している



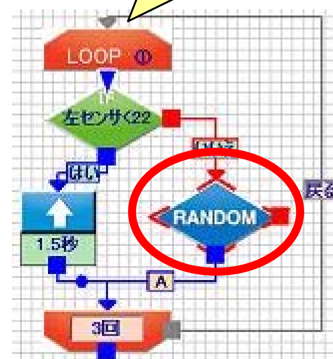
×くり返しの終了を通らず
プログラムを終了させている



×くり返しの途中で
別のくり返しに割り込む

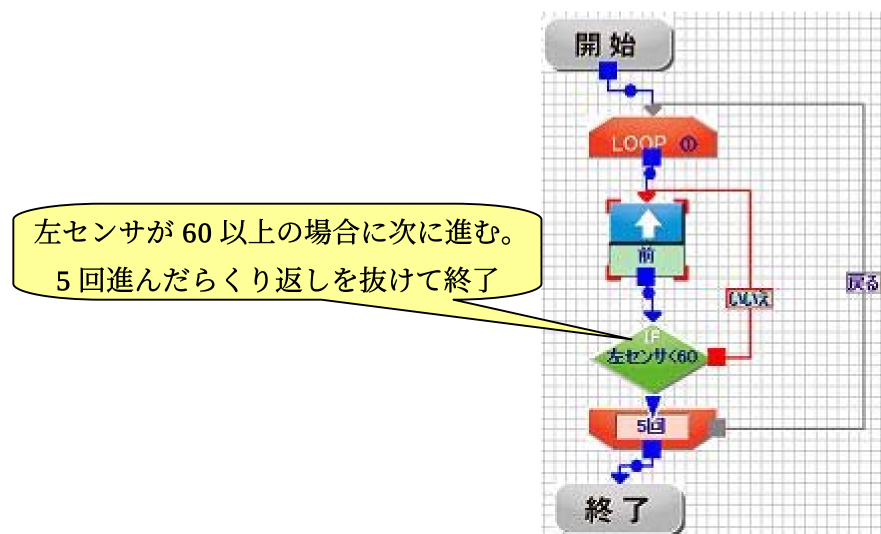


×くり返しの途中で
矢印をつなぎ忘れている



いずれの場合も、原因のポイントとなるのは、「**繰り返しが終わらないのにプログラムが終了する場合がある**」もしくは「**くり返しの開始と終了が異なる組み合わせでつながっている**」というものです。くり返しに問題がある場合、プログラムの最初から問題場所までの順序でアクションブロックの色が変わって表示されるので、それをヒントに問題を改善しましょう。

ちなみに、下図のように「くり返しの中でずっとプログラムが終わらない場合がある」という接続については、問題ありません。下図のプログラムでは「センサに 5 回反応があったら終了する」という動作をさせることができます。

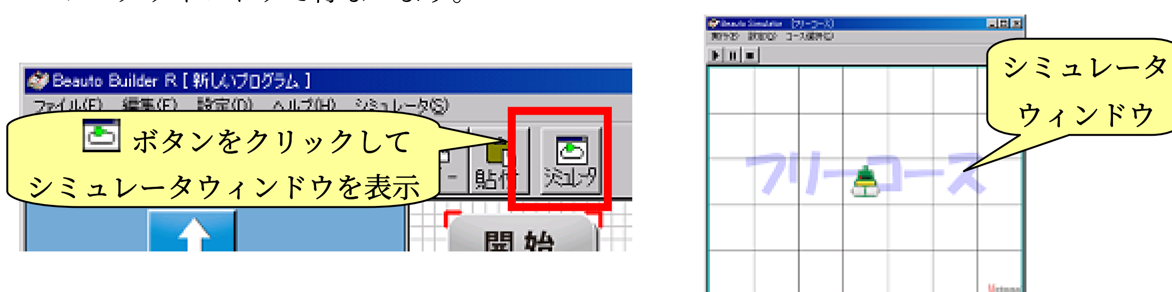


5.シミュレータについて

本ソフトウェアのシミュレータ機能を利用することで、実際にロボットがない場合もプログラミングの動作確認・学習をすることができます。なお、シミュレータと実際のロボットでは、モーターのスピードやセンサの反応などで違いがあるため、シミュレータと実際のロボットが同じプログラムで正しく動作するとは限らない点に注意してください。

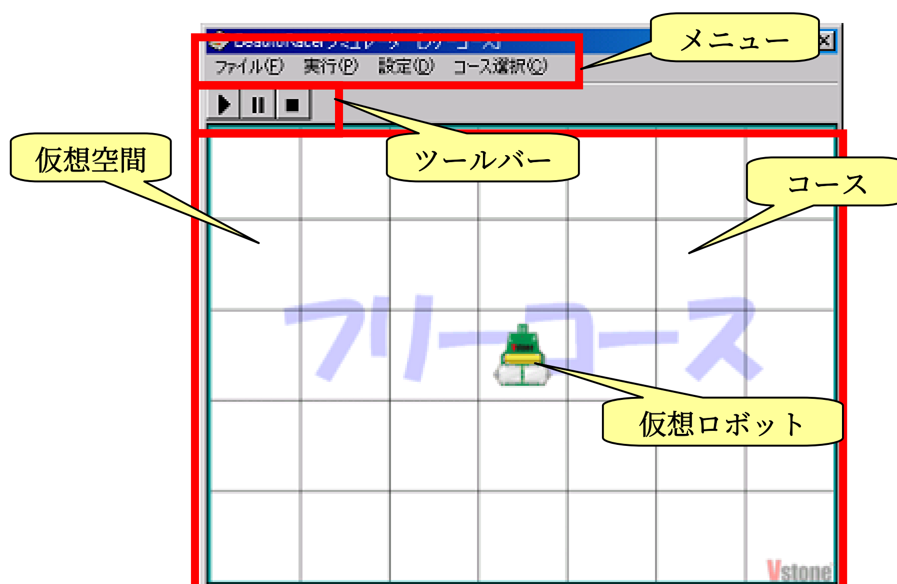
5-1.シミュレータの起動

シミュレータを起動する場合は、ツールバーのボタンをクリックしてください。クリックすると、「シミュレータウィンドウ」を別に開きます。シミュレータは、プログラミングは全て実際のロボットと同じくプログラムエリアで行ないますが、動作確認はこのシミュレータウィンドウで行ないます。

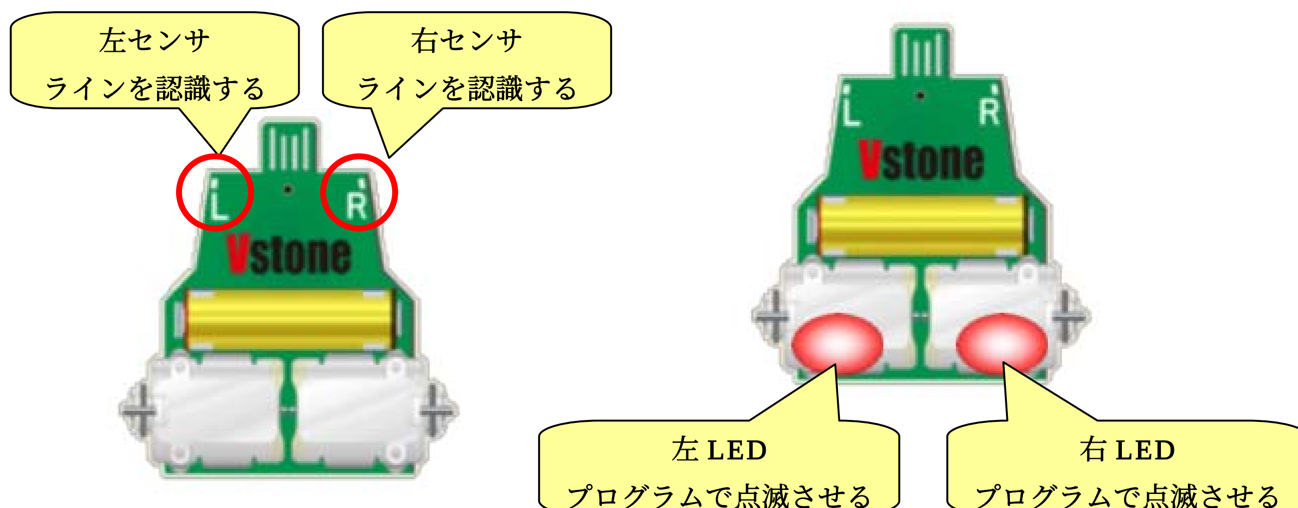


5-2.シミュレータウィンドウの画面説明

シミュレータウィンドウの画面は上下に大きく分かれます。画面上部には「メニュー」及び「ツールバー」が備わっており、これらを操作して作成したプログラムを実行したり、使用するコースを選択したりします。画面下部は「仮想空間」で、プログラムを実行する「仮想ロボット」や「コース」が表示されます。



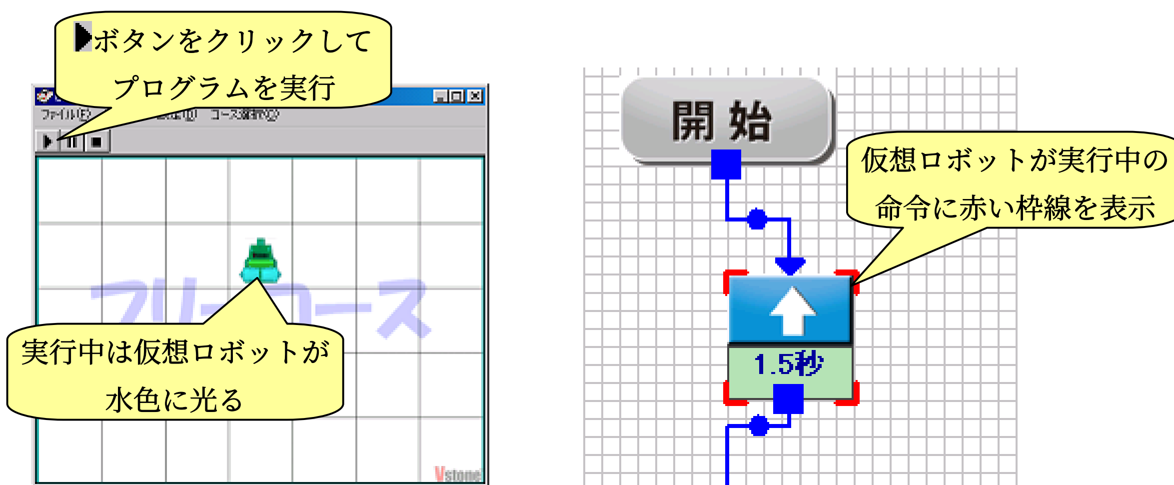
仮想ロボットには、実際の Beauto Racer と同じくモータ・センサ・LED などのインターフェースが備わっています。それぞれの搭載場所や名称は下図の通りです。



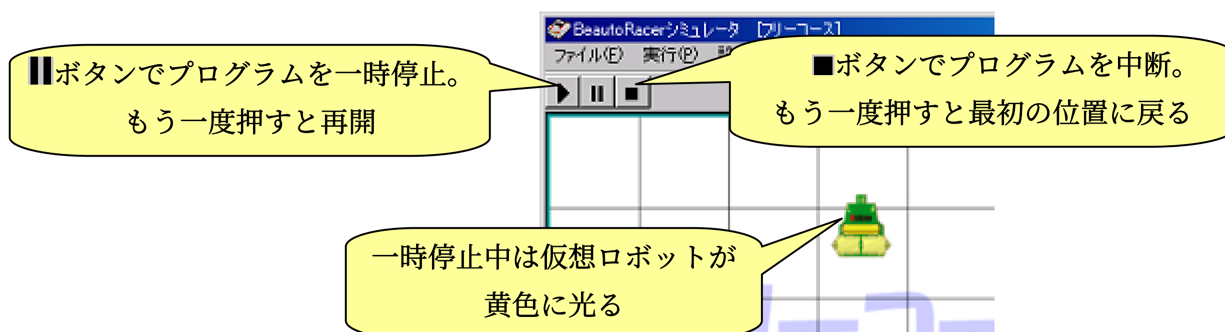
5-3.シミュレータウィンドウの操作

シミュレータからプログラムを実行する場合は、シミュレータウィンドウのツールバーより▶ボタンをクリックしてください。クリックするとロボットが水色に光り、作成したプログラムを実行します。プログラムの最後に到達するとロボットの動作が止まります。

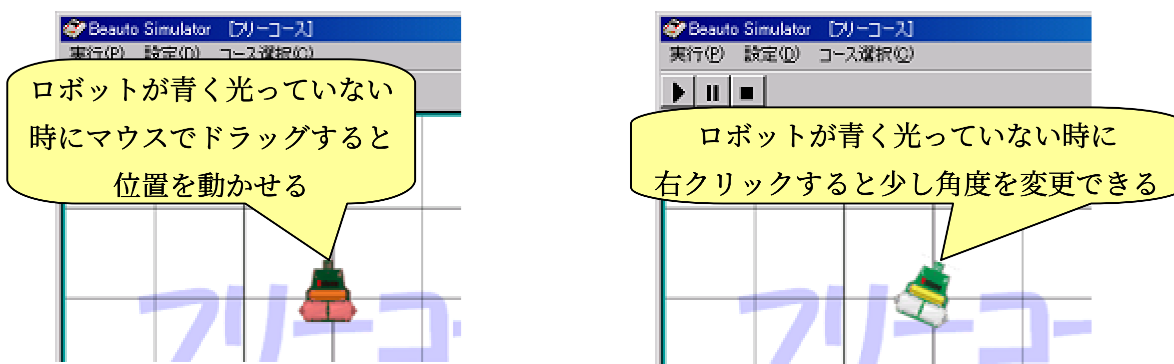
プログラムの実行中は、現在仮想ロボットが実行している命令に応じてメインウィンドウのアクションブロックの赤い枠線が切り替わります。



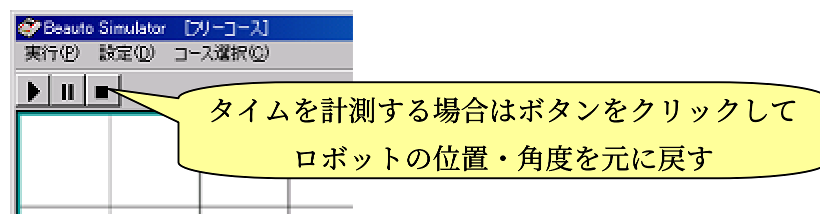
プログラムを途中で終了する場合は■ボタンを、一時停止する場合は❚❚ボタンをそれぞれクリックします。プログラムを一時停止するとロボットが黄色く光り、もう一度❚❚ボタンを押すと停止したところからプログラムを再開します。また、**プログラムを実行していない状態で■ボタンをクリックすると、コースの最初の場所にロボットが移動します。**



また、仮想ロボットが青く光っておらず、且つプログラムの実行中でない場合は、**仮想ロボットをマウスでクリック・ドラッグして位置と向きを変更することができます。**

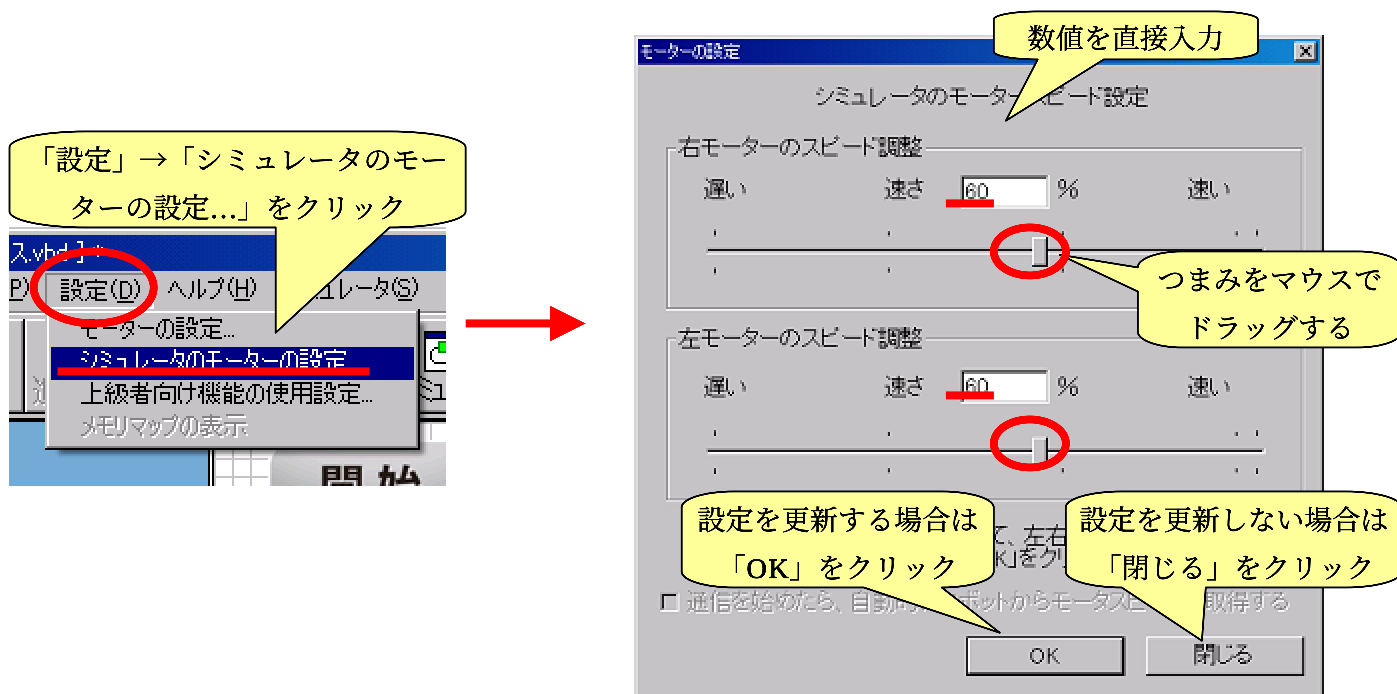


なお、**マウスでロボットの位置・向きを動かすと、スタートの条件が変わってしまうためコースのタイム計測が行なわれなくなります。**もう一度タイムを計測する場合は、■ボタンをクリックしてロボットをコースのスタート位置に戻してください。



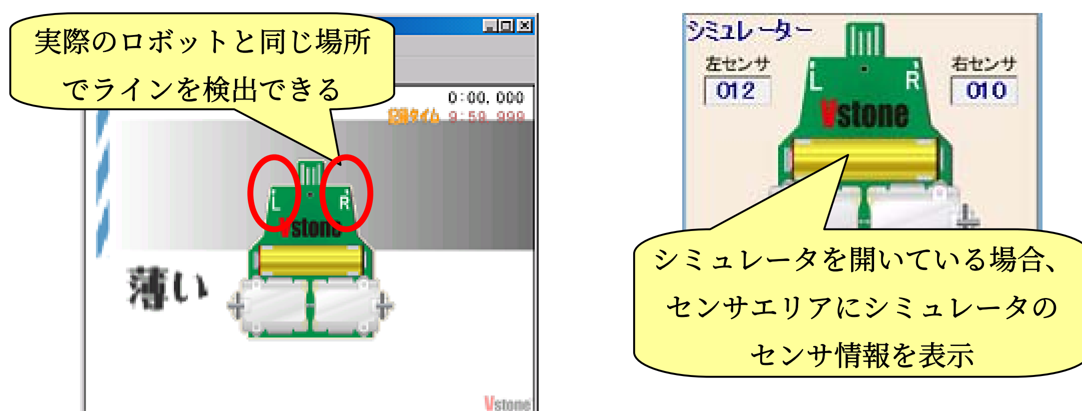
5-4.シミュレータのモータースピードの調整

シミュレータの仮想ロボットは、実際のロボットと同じくモーターのスピードを変更することができます。仮想ロボットのモータースピードを調整する場合は、メインウィンドウのメニューより「設定」→「シミュレータのモーターの設定…」をクリックしてください。クリックすると、設定を行なうダイアログを開きます。



5-5.シミュレータのセンサ情報

シミュレータウィンドウを開いている場合、センサエリアにシミュレータのセンサ情報を表示します。仮想ロボット前方の「L」「R」の部分には、コースの色の濃さを感知するセンサが備わっています。これらの文字のちょうど真下付近が感知する場所になります。センサの数値は、白い場所で 10 付近、黒い場所で 160 付近になります。



5-6.シミュレータウィンドウのメニューについて

シミュレータウィンドウには独自のメニュー・ツールバーが備わっており、プログラムの実行やコースの選択、画面の表示方法などをこちらで変更できます。それぞれの項目の機能は下記のとおりになります。

シミュレータウィンドウのメニュー・ツールバー

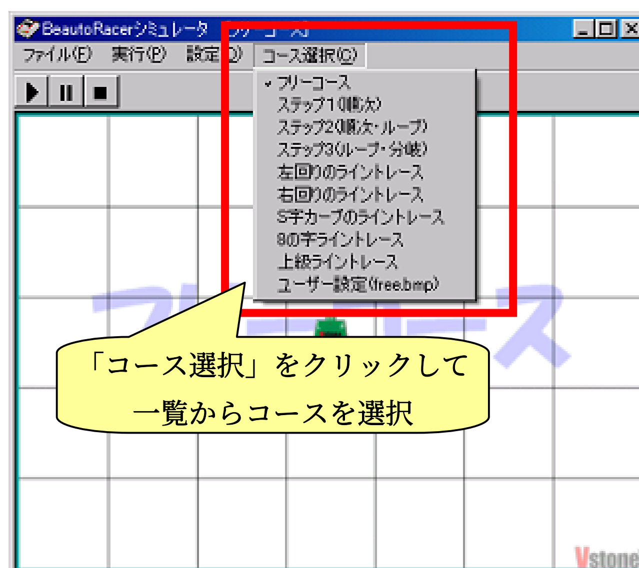
- 実行
  - 再生(▶)・・・・・作成したプログラムを実行します。一時停止中に選択するとプログラムの実行を再開します。
  - 停止(■)・・・・・実行中のプログラムを中断します。また、プログラムを実行していないときに選択すると、仮想ロボットをコースの最初の位置に戻します
  - 一時停止(❚❚)・・・実行中のプログラムを一時停止します。一時停止中に選択するとプログラムの実行を再開します。
- 設定
  - ズームアウト・・・・・シミュレータウィンドウをコース全体が見える視点に切り替えます。
  - ズームイン(2 倍)・・・・シミュレータウィンドウを、仮想ロボットを中心に約 2 倍の大きさに拡大します。
  - ズームイン(4 倍)・・・・シミュレータウィンドウを、仮想ロボットを中心に約 4 倍の大きさに拡大します。
  - 標準速度・・・・・・・プログラムの実行速度を通常の速度に戻します。
  - スロー再生・・・・・・プログラムの実行速度を通常より遅く設定します。
  - もっとスロー再生・・・プログラムの実行速度を、スロー再生より遅く設定します。
  - 記録タイム確認・・・・下記のダイアログを表示し、各コースの最速タイムの記録を表示します。ダイアログの「記録をリセット」ボタンをクリックすると、現在の記録を削除し初期状態に戻します。

- コース選択
  - 現在登録されているコースから、ロボットを走らせるコースを選択します。

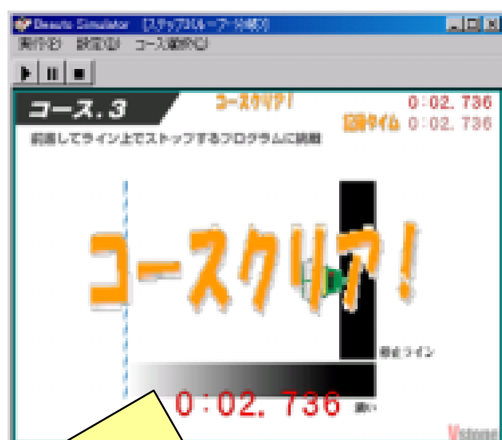
5-7.コースの選択

シミュレータには、実践課題などを含めて 10 種類のコースを選択できます。コースの色に対してシミュレータのセンサが反応するようになっており、ライントレースなどのプログラミングを作成できます。また、実践課題のコースでは、課題の達成を「センサが使われているか」「正しい経路でロボットが移動しているか」などで判断したり、課題を達成した際の最速時間を記録することができます。ちなみに、最初に表示されるコースは、プログラムの動作確認用のため、このような機能はありません。

コースを変更する場合は、シミュレータウィンドウのメニューより「コース選択」をクリックします。クリックするとコースの一覧を表示するので、好きなコースを選んでクリックしてください。



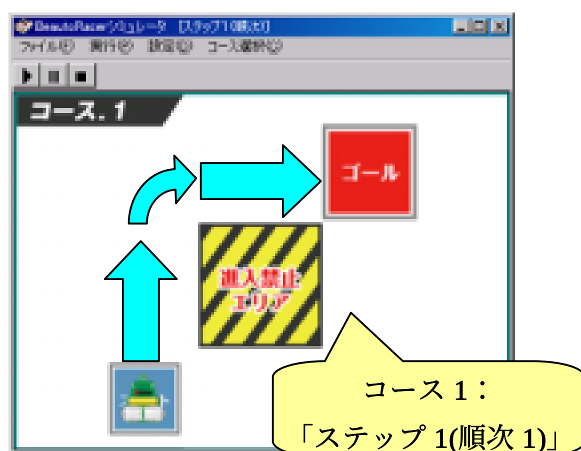
一部のコースを除いて、それぞれのコースには課題が決められており、プログラムを実行したときに課題が成功・失敗したかの結果を表示します。それぞれの課題は、「2.単純なプログラムの作成」「3.センサを使うプログラムの作成」「4.くり返しを使うプログラムの作成」の説明を応用すれば成功できるようになっています。これらの項目と下記の解説を元に、課題成功に挑戦してみてください。

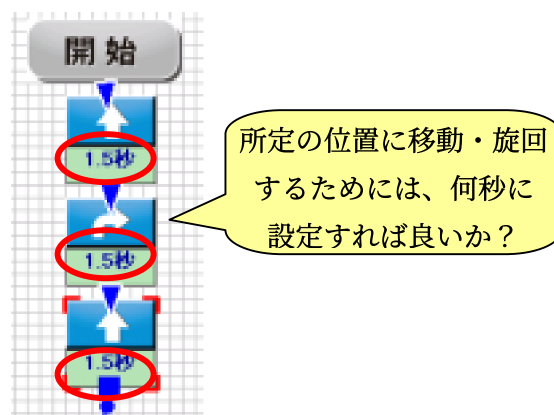




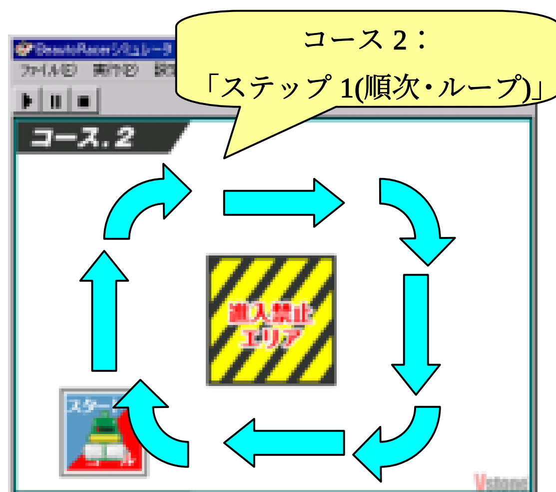
5-7-1.順次処理の課題挑戦（コース１・コース２）

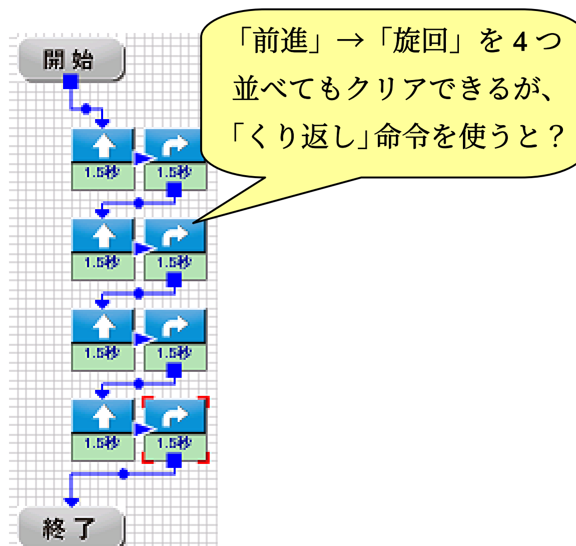
コース１は、画面左下のスタートから右上のゴールに移動することが目標です。ただし、中央に進入禁止エリアがあるため、そこを避けて「前進」→「90 度旋回」→「前進」という順番で進む必要があります。ただし、正しい位置で停止・旋回できるように、それぞれの命令の秒数を設定する必要があります。





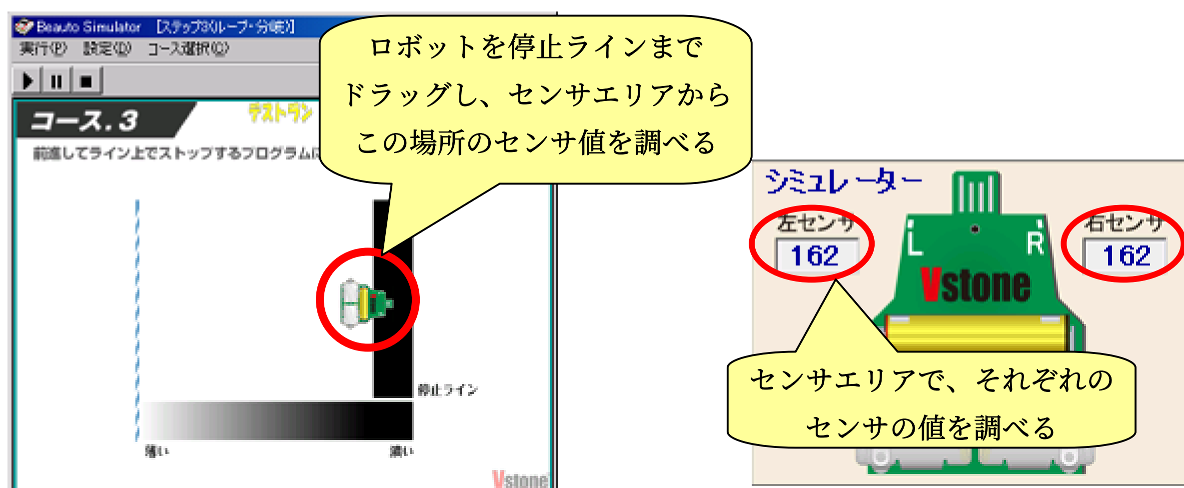
コース２はコース１を発展した内容で、プログラムをスタートしたら、進入禁止エリアの周囲を回り戻ってくるコースです。「前進」→「90 度旋回」を４回実行するとクリアできます。同じ命令を４つ並べてもプログラムは完成しますが、ここでは「くり返し」の命令を使用してクリアを目指しましょう。



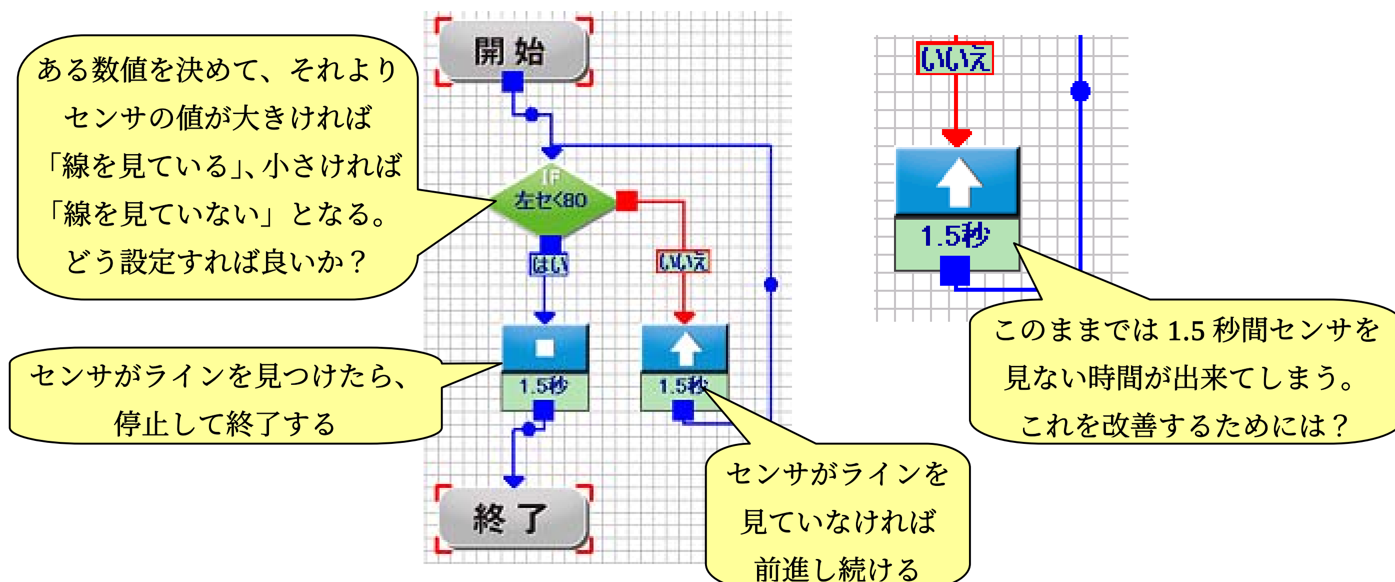


5-7-2.センサを使う（コース 3）

コース 3 は、センサを使った分岐を学ぶことが目的です。ロボットが分岐を行なうためには、ロボットに搭載されたセンサを使って状況をしらべる必要があります。**仮想ロボットをマウスでドラッグし、コース右側の停止ラインに動かしましょう。動かしたらセンサエリアで値を確認し、白い場所と黒い場所でどの程度数値に差があるか調べて見ましょう。**

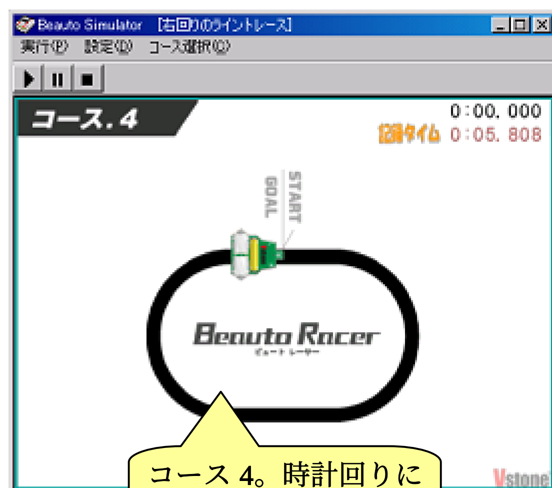


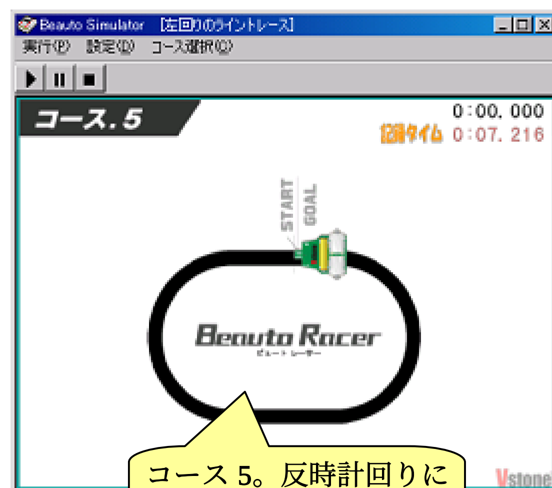
調べた値を元に、**「センサがラインを見ていれば停止」「センサがラインを見ていなければ前進」**というプログラムを作成してみましょう。分岐の命令については「3-2.分岐の使い方」で基本的な使い方を説明しているので、そちらも参考にしてみましょう。ポイントは、分岐の命令で「入力する数値」と「分岐の方法」をどう設定するか、また、「はい」「いいえ」の二つの分岐の矢印をどのように接続するかになります。他にも、「前進し続けるには矢印の接続をどうすれば良いか」「前進中にセンサの変化を取りこぼさないためにどうすれば良いか」という課題もあります。下の図とヒントを参考に、正しいプログラムを作成してみましょう。

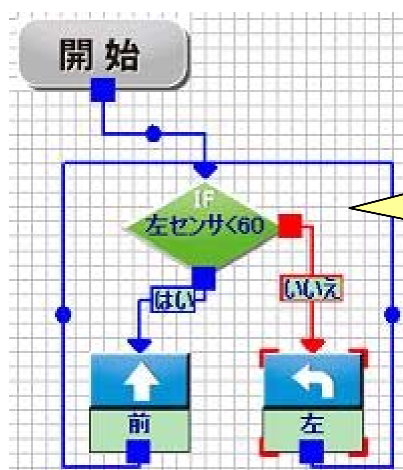


5-7-3.センサを一つ使ったライントレース（コース 4,5）

コース 4,5 は、センサを一つ使用してライントレースすることが目標です。コース 4 は時計回り、コース 5 は反時計回りに一周させます。センサを一つ使ったライントレースは「3-3. ライントレースのプログラミング」で解説しています。前述のコース 3 の解説と合わせて、それぞれのコースでどのようなプログラムを作れば良いか、またどちらのセンサを使えば良いか、良く考えてプログラミングしましょう。



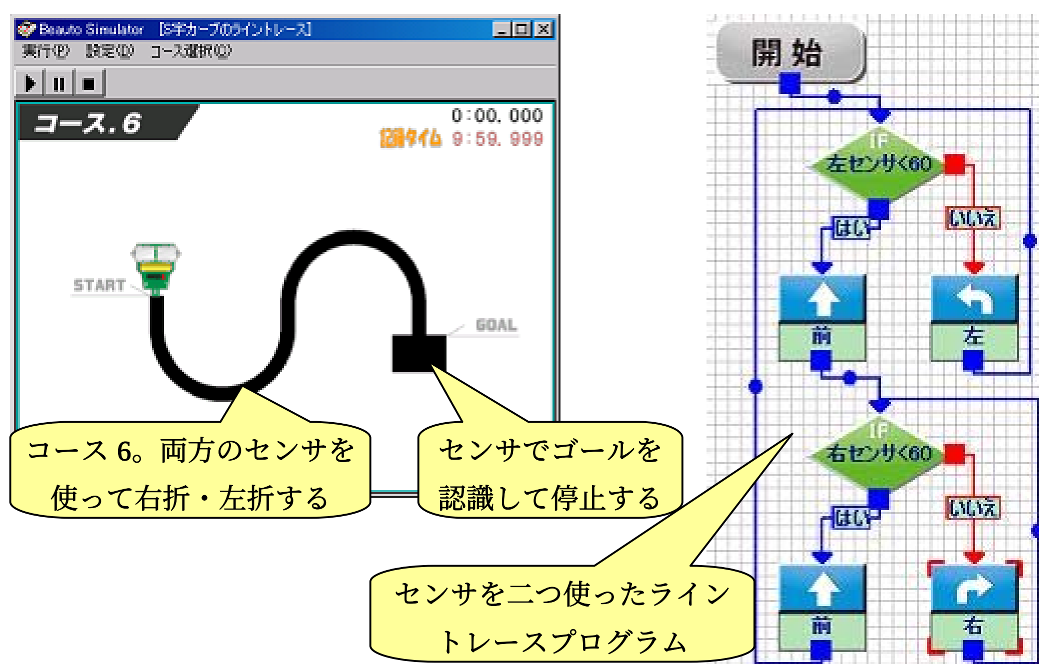




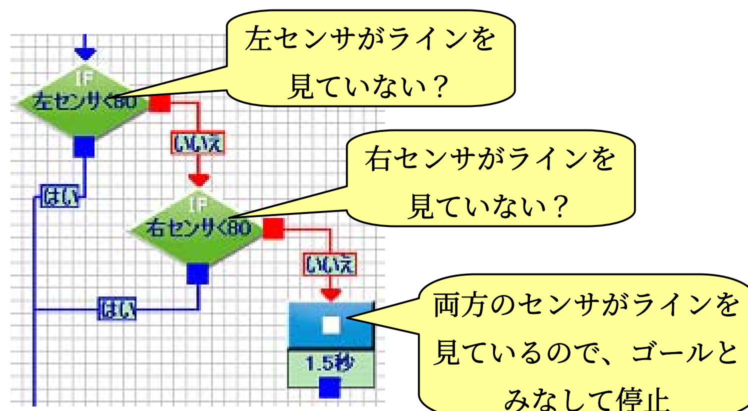
5-7-4.センサを二つ使ったライントレース（コース 6）

コース 6 は、S 字カーブをライントレースしてゴールエリアで停止することが目標です。このコースは、コース 4,5 と異なり一つのコースで右折･左折がどちらも出てくるため、両方のセンサを使用する必要があります。また、最後にゴールエリアで止まらないと課題成功にならないため、これもセンサを使ってゴールエリアを認識する必要があります。

一度に大きく難易度が上がったように思えますが、プログラムの大半は「3-4.左右センサを使ったライントレース」を参考に作成できます。

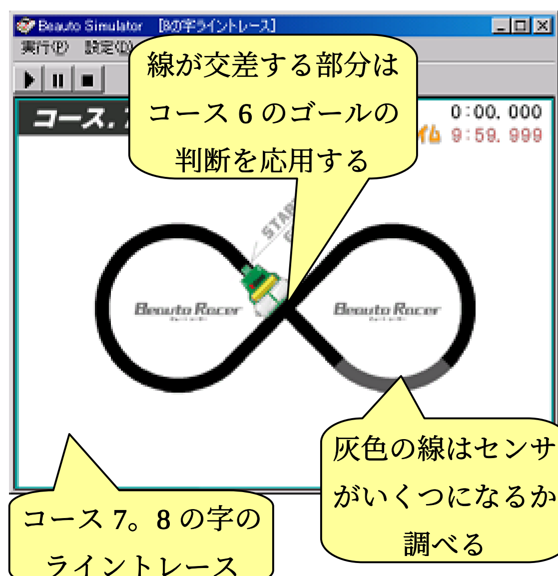


ゴールエリアの認識は「センサが両方ともライン（黒い部分）を見ているか」で判断できます。分岐命令では一度に二つのセンサを使った判断が出来ないため、下図のように分岐命令を連続で並べます。ライントレースのプログラムに下図のプログラムをどう組み込めばよいか考えて見ましょう。ヒントは、既にライントレースのプログラムに組み込まれた分岐が、「二つのセンサを使った判断の最初の分岐として使える」という点です。



5-7-5.8 の字のライントレース・上級ライントレース（コース 7,8）

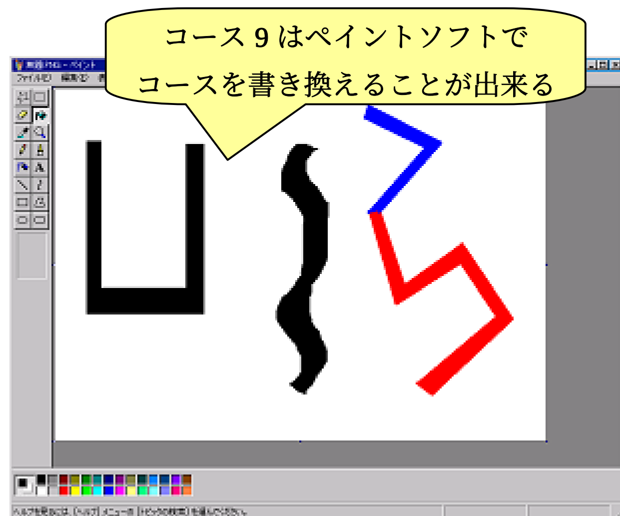
残り二つのコースは、上級者向けの難しいコースです。これまでのコースにはなかった点として「ラインが交差する」「灰色のラインが出てくる」があります。前者はコース 6 の「ゴールエリアに到着したか」を判断する処理を応用して、「両方のセンサがラインを見ているときは前進する」というものに変更すれば成功します。後者は仮想ロボットをマウスでドラッグして、灰色の部分でセンサの数値がいくつになるのか調べる必要があります。

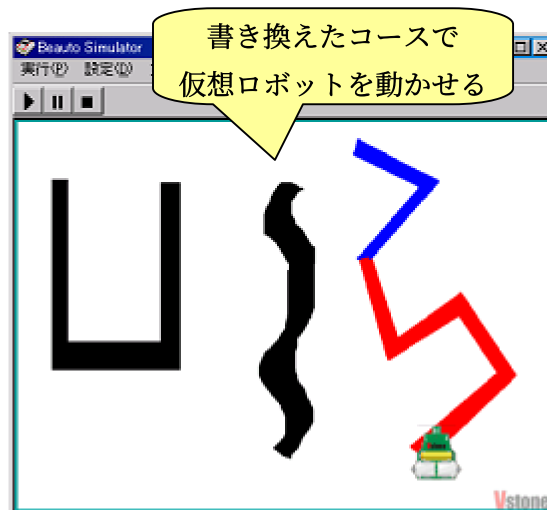




5-7-6.オリジナルコース（コース 9）

コース選択の一番下にある「ユーザー設定(free.bmp)」は、地面に何も書かれていないコースですが、ペイントソフトなどを使ってコースの元になる画像データを書き換えることで、新しいコースを作成することができます。





コース 9 の元となる画像ファイルは、本ソフトウェアのフォルダに含まれる「texture」フォルダの「free.bmp」というファイルになります。こちらを、Windows に標準で付属する「ペイント」などで開いて、線などを書き加えて保存し本ソフトウェアを起動すると、コースが書き換えた画像に変わります。ただし、**他のコースのように課題を与えて、それに対して成功・失敗を判断させることは出来ません。**

画像の変更やファイルへ保存する際には必ず以下の点にご注意ください。**これらが正しくない場合、シミュレータ機能を利用できなくなる場合があります。**また、ファイルを書き換える前に、**必ずバックアップをとってください。**

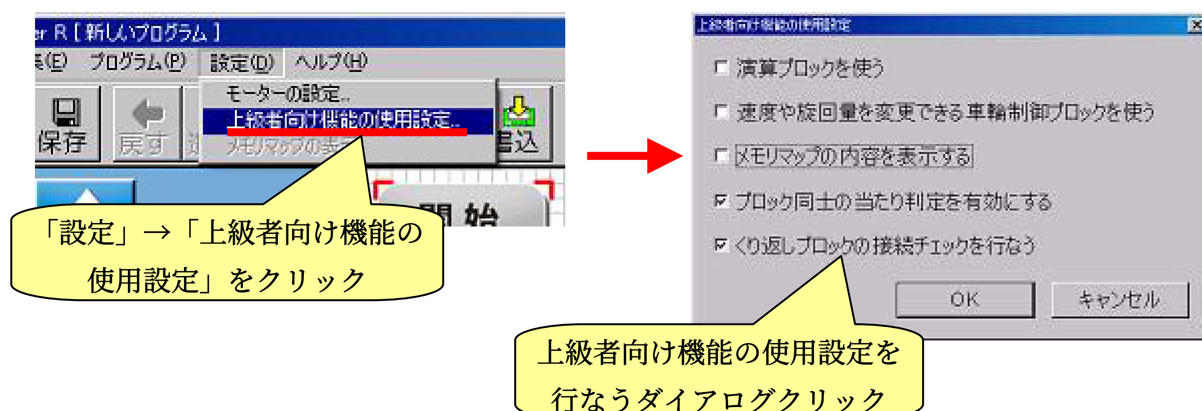
- **ファイル名を変更しない**
- **ファイルの形式を「24 ビットビットマップ(*.bmp,*,dib)」から変更しない**
- **画像サイズを 700×500 ピクセルから変更しない**

画像の変更については、使用する色や図の形状に特に制限はありませんが、ライントレース用の線を書く場合は「線の太さをなるべく太く」、また「線の色をなるべく黒く」するようにしてください。また、画像の 1 ピクセルが約 1mm に相当するので、自分が実際に紙で作ったコースなどを再現させることも可能です。

6.上級者向けの機能について

本ソフトウェアには、より多様なプログラミングができるように、変数やメモリマップなどを利用したプログラミング上級者向けの機能が備わっています。簡単なプログラムであればこれまで説明した内容でも充分作成できますが、上級者向けの機能により、プログラミングをより深く理解できます。

上級者向けの機能は、本ソフトウェアをインストールした状態では利用できない設定になっています。機能を利用できるようにする場合は、設定ダイアログを表示して設定を切り替える必要があります。上級者向けの機能の設定ダイアログは、メニューより「設定」→「上級者向け機能の使用設定…」をクリックすると開くことができます。

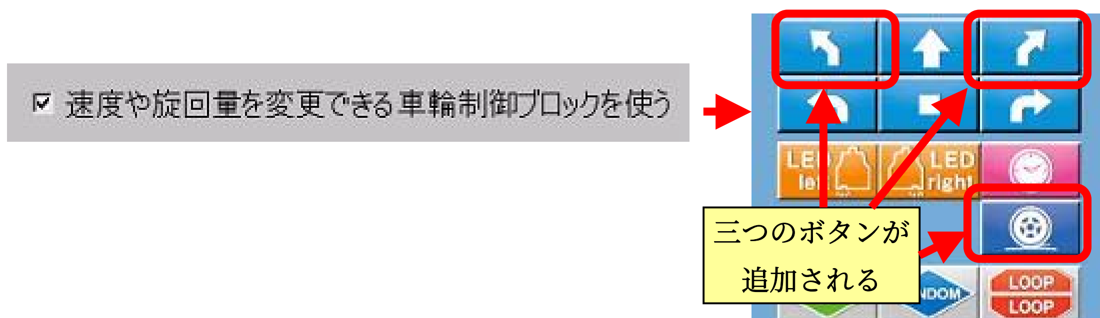


ダイアログでは 5 つの設定項目が表示されており、項目をクリックしてチェックを切り替えることで設定を変更します。ダイアログの「OK」をクリックすると変更した設定を適用します。それでは、各設定項目の詳細について説明します。

6-1.速度や旋回量を変更できる車輪制御について

まずは扱いが簡単なものから説明します。「速度や旋回量を変更できる車輪制御ブロックを使う」にチェックを入れると、モーターの速度を自由に変更できる命令を使用できるようになります。最初に使用できる「前進」「旋回」などのモーターの命令は、モーターの回転速度が常に一定でした。この速度は、モータースピードの設定を行なうことで変更できますが、それでも一つのプログラムに存在するモーターの命令は、必ず同じ速度で動作します。それに対して、この機能を有効にすると、一つのプログラムで、命令ごとにモーターの速度を自由に変更できるようになります。

「速度や旋回量を変更できる車輪制御ブロックを使う」にチェックを入れて、ダイアログの「OK」をクリックすると、アイコンエリアに下図のボタンが追加されます。

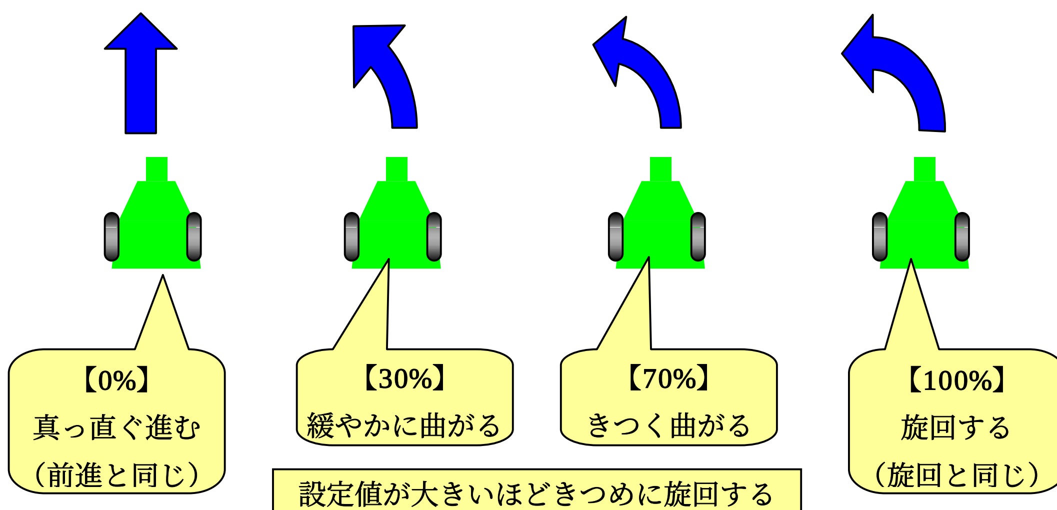


下図の 2 つのボタンは、「右前に進む」と「左前に進む」の命令になります。これらは「左右旋回」の命令を拡張し、**「どのくらいの角度で旋回するか」を設定できる命令**です。これらの命令における設定エリアでの設定内容は右下図のようになります。大部分は従来のモーターの命令と同じですが、**「??%」という設定が追加されています。**

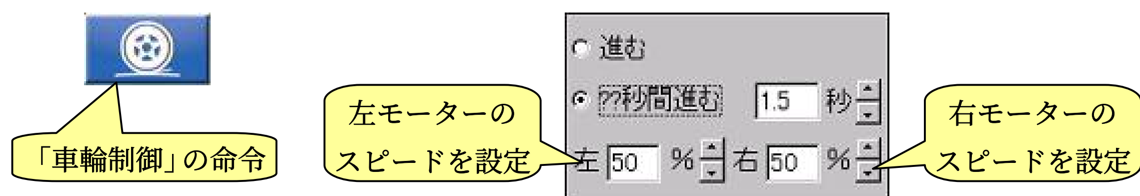


「??%」の設定には 0~100 の間で数値を入力します。この数値は「小さいほどロボットが真っ直ぐ進む」また「大きいほどロボットがきつく旋回する」ようになっています。

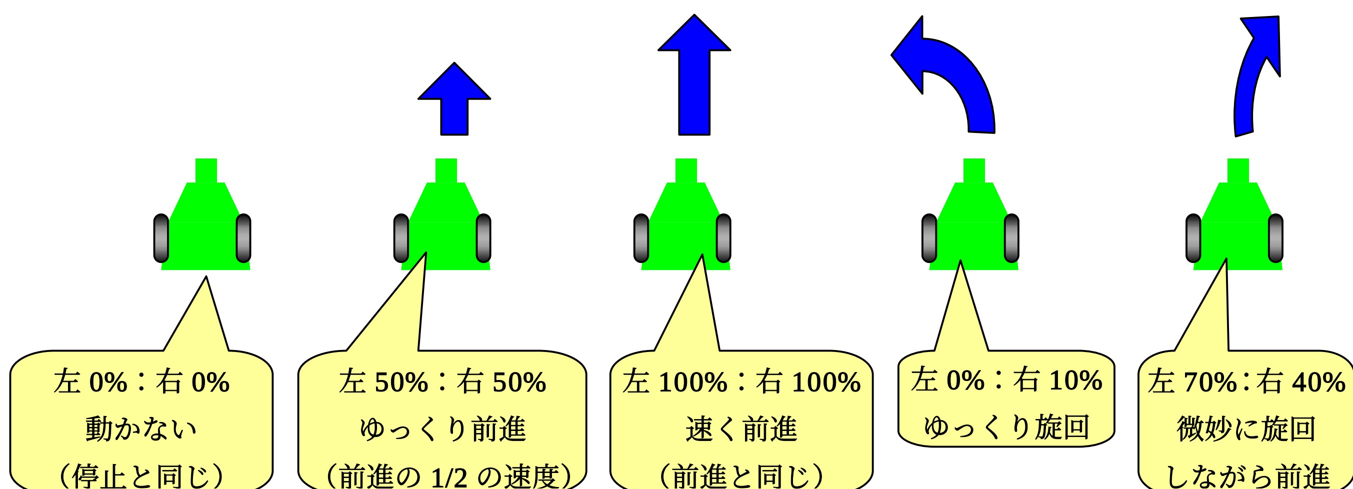
ちなみに、他の設定項目については従来のモーターの命令とまったく同じです。「??秒間進む」であれば指定の時間だけ移動し、「左前に進む」「右前に進む」であれば、モーターを回しっぱなしで次の命令に進みます。



下図のボタンは「車輪制御」の命令になります。こちらは「右前に進む」と「左前に進む」を拡張し、**左右両方のモーターを好きな速度で動かせる命令**になります。この命令の設定内容を設定エリアで確認すると、右下図のようになります。



車輪制御の命令にも**「??%」の設定がありますが、こちらは左右モーターのスピード**になります。数値は 0~100 の範囲で設定し、**数値が大きいほどスピードが速くなります**。左右のモータースピードの数値をそろえることで、任意の速さで前進させたり、任意の速度で「右前に進む」「左前に進む」のように柔軟な角度の旋回をさせたりすることができます。



6-2.メモリマップの表示について

「メモリマップの内容を表示する」にチェックを入れると、ロボットの内部的な数値である「メモリマップ」を下図のように画面に表示します。

「**メモリマップ**」とは、**ロボットのマイコン内部に記録されている様々な数値**を表します。ロボットはプログラムに登場する様々な数値を、マイコンの中に個別に記録しています。これまでに登場した「モータースピード」「左右センサ値」や、「LEDのON/OFFの状態」も数値として記録されます。これらの数値はロボットの動作中やプログラム実行中に、ロボットによって適時書き換えられます。メモリマップを表示させた状態でロボットとPCを通信させると、ロボットから定期的に最新のメモリマップの数値を読み込み、画面上のメモリマップの表示を更新します。

また、後述の「**演算**」の命令を使うことで、**プログラムからメモリマップ中の好きな項目に数値を書き込むことができます**。数値を書き込むと、**書き込んだ項目によってロボットのモーターやLEDが動作するなどの変化が現れます**。また、一部の項目はロボットだけが数値を書き換えることがというものもあります。メモリマップの各項目の意味は次の通りです。

「**モーターの速度**」は、現在のモーターのスピードを表します。**数値が大きいほどモーターが速く動き、0の場合はモーターが動きません**。プログラム実行中に前進などの命令が出てきた場合、ロボットがこの数値を書き換えてモーターを動かします。また、プログラムから好きな数値に書き換えても、モーターを動かすことができます。

「**モーターのゲイン**」はモータースピードを表します。**数値の意味はモータースピードの設定と同じですが、ここでは 0~255 の範囲で数値が表示されます。この数値は「モータースピードの設定」でのみ変更可能で、プログラムから書き換えることができません。**

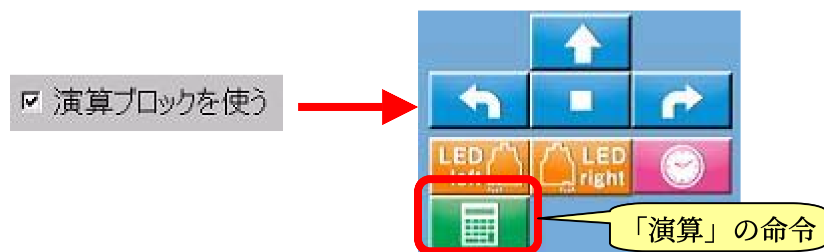
「**左センサ**」「**右センサ**」及び「**センサ③〜⑥**」は、ロボットのセンサの数値です。ロボットにはセンサが 2 つしかありませんが、別途拡張することであと 4 つセンサを増やすことができます。その際「センサ③〜⑥」に拡張したセンサの値が表示されます。センサに何も接続されていない場合でも、入力が不安定になり数値が勝手に書き換わる場合があります。また、**この数値は常にロボットが新しい値に書き換えるため、プログラムから数値を書き込めません。**

「**LED**」は、左右 LED の ON/OFF の状態を表します。**ON/OFF の状態は 0~3 までの数値**で表され、「**0 の場合は左右 LED ともに OFF**」「**1 の場合は左 LED のみ ON**」「**2 の場合は右 LED のみ ON**」「**3 の場合は左右 LED ともに ON**」となります。この項目も、ロボットが自分で数値を書き換えたり、プログラムで好きな数値に書き換えたりすることができます。

「**変数 a~h**」はプログラムで使用できる変数を表します。「**変数」とは、ユーザがプログラムで使用する数値を自由に読み書きできるための項目**で、ロボットが勝手に内容を書き換えることはありません。使用例として、「変数 a〜h の順番で 1 秒ごとにセンサの値を記録する」というプログラムを作成すれば、過去数秒間のセンサの数値をロボットに記録しておくことができます。

6-3.演算ブロックの使用について

「演算ブロックを使う」にチェックを入れると、メモリマップの書き換えを行なう「演算」の命令を使用できるようになります。この項目にチェックを入れると、アイコンエリアに下図のボタンが追加されます。



「演算」の命令はプログラムエリアで下図のように表示されます。また、設定エリアでの表示は右下図のようになります。

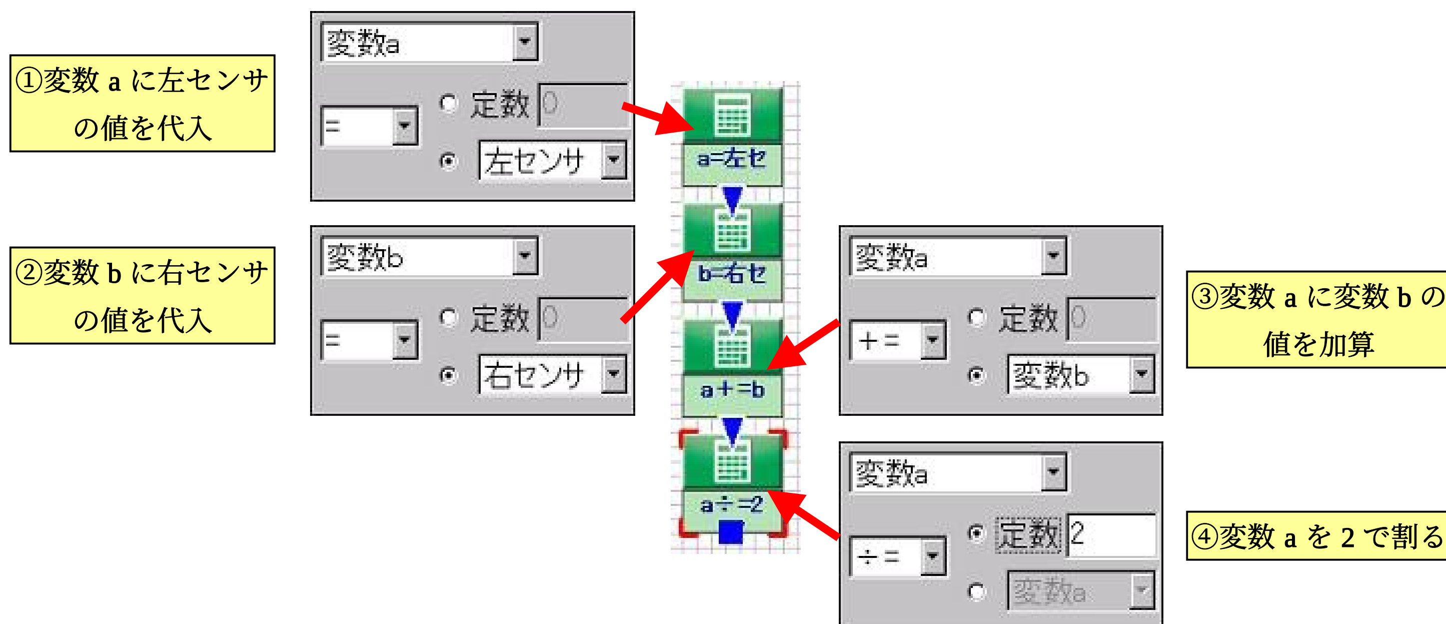


設定エリアの項目は数が多いので、それぞれ説明していきます。まず **1 行目は、数値を代入するメモリマップの項目を選択**します。ここでは、「変数 a~h」「LED」「左右モーターの速度」から選択できます。

2 行目左側は、メモリマップへの数値の代入方法を選択します。演算の命令では、単純に決められた数値に書き換えるだけでなく、**選択したメモリマップの項目に現在入っている値に対して「足す」「引く」「掛ける」「割る」の四則演算を行なうこともできます。**



演算命令の具体的な使い方について、下記にプログラム例を掲載します。このプログラムは「**左右センサの平均値を計算する**」というものです。左右センサの平均値は「 **(右センサ値 + 左センサ値) ÷ 2** 」という計算で求められますが、演算の命令は一度にこれだけの計算をすることができません。また、メモリマップの部分で説明した通り「左右センサ」の項目は数値を書き換えることができません。そこで、**左右のセンサの値を変数 a,b に一旦代入**します。**続いて変数 a に変数 b を加算**し「(右センサ値 + 左センサ値)」の値を求めます。**最後に変数 a を 2 で割り、平均を求めます。**



なお、変数の数値は必ず「0～255」の整数に収まり、小数や負の数はありません。計算結果がこれらの数になる場合、小数点以下の数は切り捨てられ、-1 以下、及び 256 以上の数は、それぞれ 0、255 に補正されます。

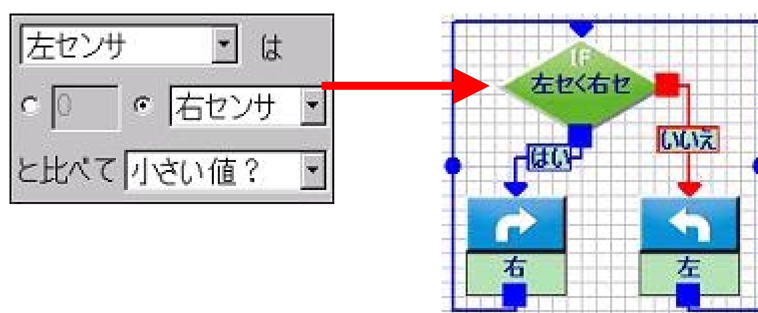
6-4.上級者向けの分岐命令について

「演算ブロックを使う」にチェックを入れると、分岐命令の設定内容も若干変化します。この機能を有効にして、設定エリアで分岐命令の設定内容を確認すると、下図のようになります。

大きな変更箇所は「1 行目でセンサ以外を選択できる」及び「比較する数値に、0~255 の数値以外にメモリマップの項目を選択できる」という点です。

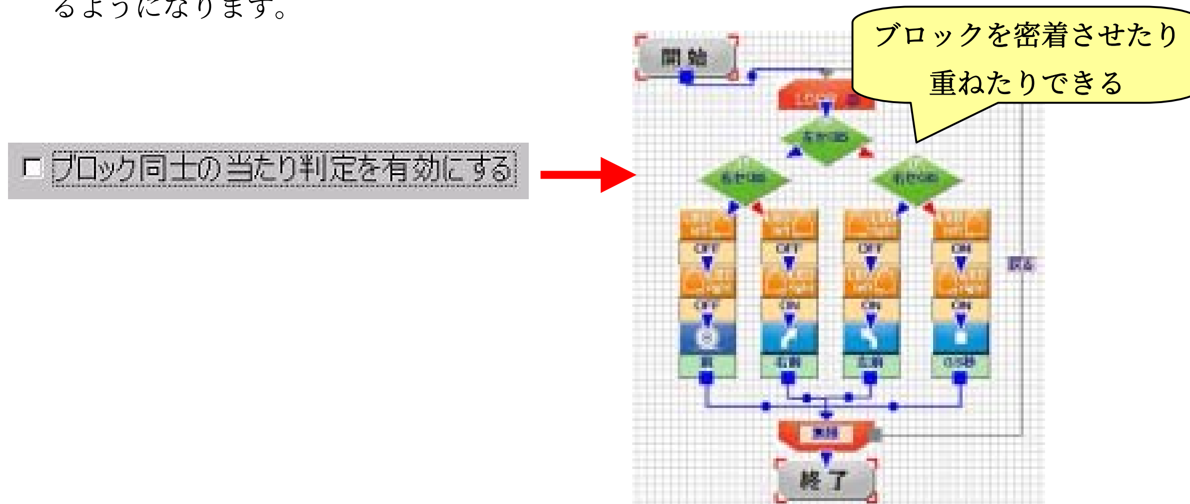


例えば下図のプログラムの場合、左右のセンサの数値を比べて、数値が大きい方のモーターを回すようになります。



6-5.ブロック同士の当たり判定について

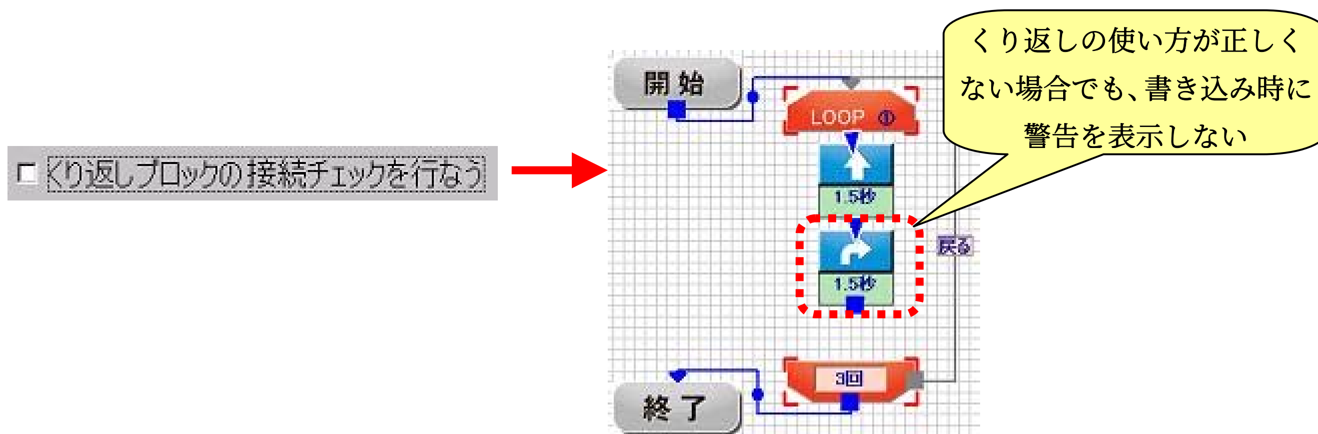
「ブロック同士の当たり判定を有効にする」の項目は、プログラムエリアのアクションブロック同士が重なり合わないようにするかどうかを設定します。この項目には最初からチェックが入っており、チェックを外すことでアクションブロック同士を重ねて配置できるようになります。



アクションブロックを大量に使用するプログラムを作成する場合は、この項目を無効にすることでプログラミングしやすくなります。

6-6.くり返しの接続チェックについて

「くり返しブロックの接続チェックを行なう」の項目は、プログラムエリアのくり返しのアクションブロックについて、矢印の接続が間違ったプログラムを書き込むときに警告を表示するかどうかを設定します。この項目には最初からチェックが入っており、チェックを外すことで、くり返しの命令の矢印が正しく接続されていなくても、書き込み時に警告を表示しなくなります。

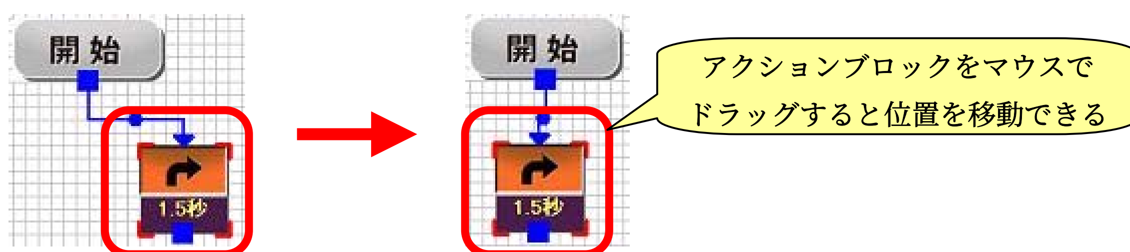


7.プログラミングに便利な機能

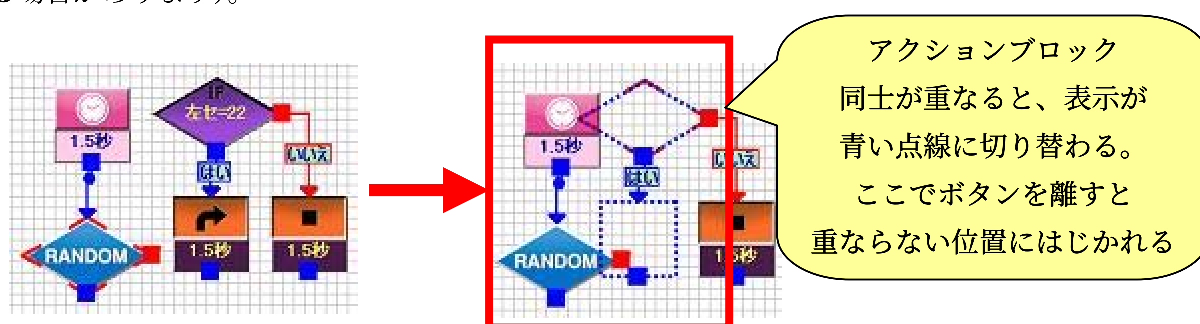
アクションブロックの移動・削除・コピーなどの操作や、メニュー・ツールバーの機能説明など、本ソフトウェアの操作方法について、以下に説明をまとめます。いくつかの説明は途中で一度紹介しているものですが、改めて確認してみてください。

7-1.アクションブロックの移動

アクションブロックの場所を移動したい場合は、アクションブロックをマウスでクリックしてドラックしてください。なお、アクションブロックは背景のマス目に沿って移動します。

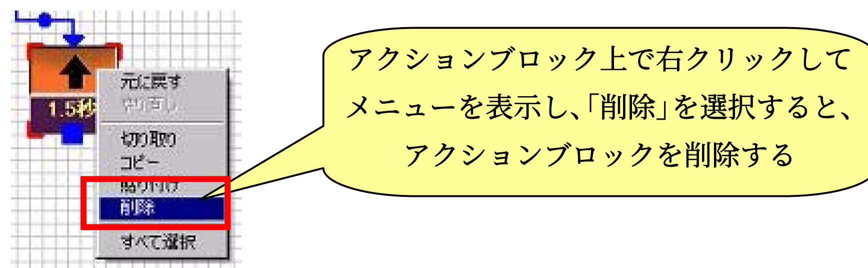


また、アクションブロックの移動中に別のアクションブロックと重なり合うと、下図のように表示が青い点線になります。アクションブロックは重ねて置くことができないため、ここでマウスのボタンを離すとブロック同士が重ならない位置まではじかれます（ただし、プログラムエリアの端や、アクションブロックが多数存在する場合などは、重なりが発生する場合があります）。



7-2.アクションブロックの削除

アクションブロックをプログラムエリアから削除する場合は、マウスカーソルをアクションブロックに合わせて右クリックします。クリックするとメニューを表示するので、メニューより「削除」を選択してください。

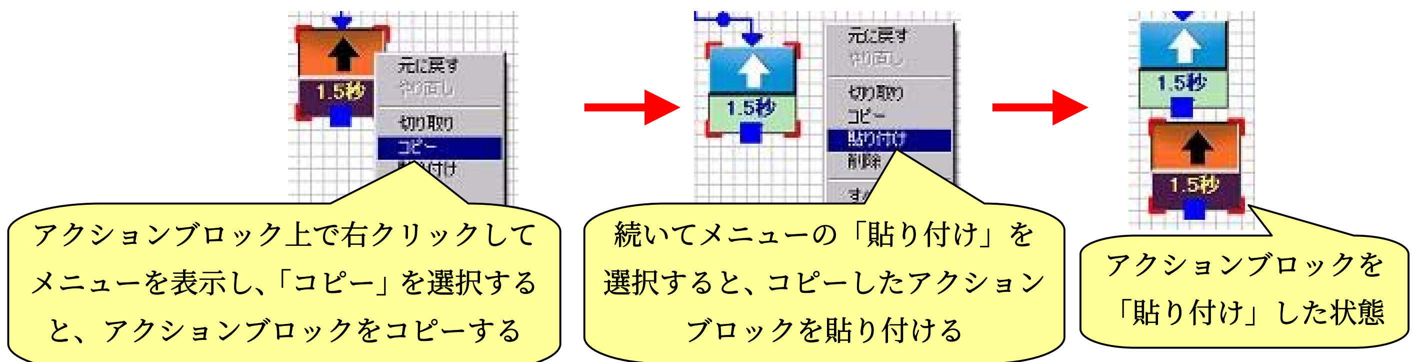


また、削除したいアクションブロックをクリックしてキーボードの Delete キーを押しても、アクションブロックを削除することができます。

7-3.アクションブロックのコピー・切り取り・貼り付け

アクションブロックを右クリックして表示されるメニューより「コピー」を選択すると、右クリックしたアクションブロックをコピーすることができます。また、メニューより「切り取り」を選択すると、右クリックしたアクションブロックをコピーしてから削除することができます。

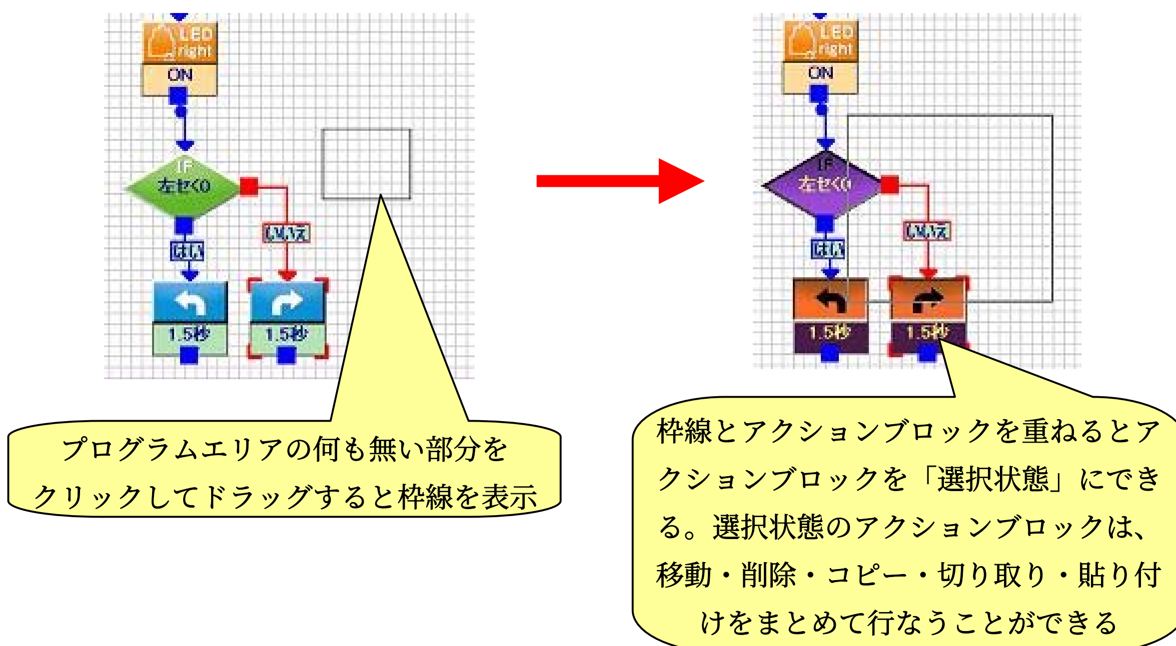
アクションブロックをコピーしてから、プログラムエリアを右クリックして表示されるメニューより「貼り付け」を選択することで、コピーしたアクションブロックをプログラムエリアに貼り付けることができます。



※なお、「開始」「終了」のアクションブロックについては、コピー・切り取り・貼り付けを行なうことができません。また、くり返しのアクションブロックは 7 個までしか貼り付けできません。

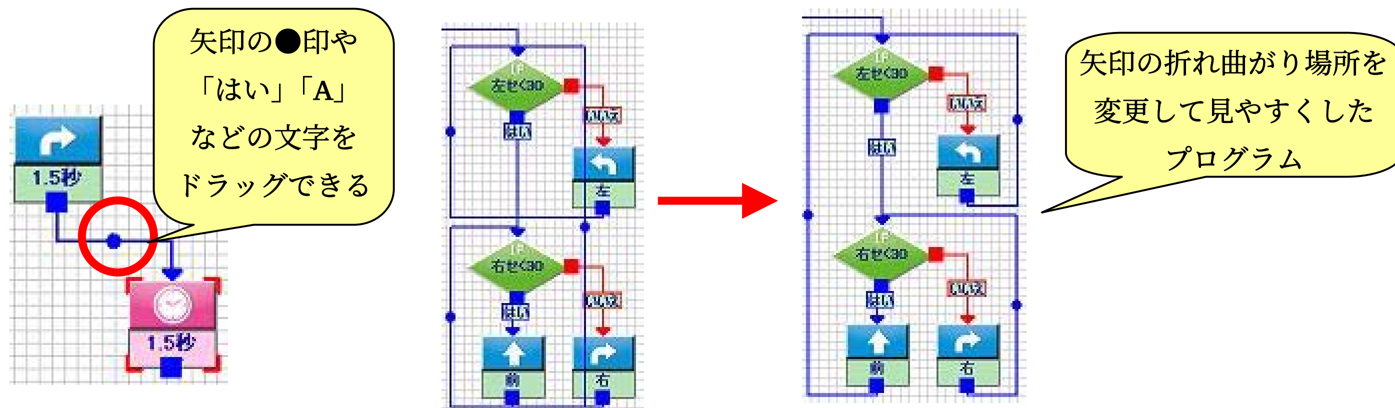
7-4.アクションブロックの選択

プログラムエリアのアクションブロックが無い部分をクリックしてドラッグすると、下図のように枠線を表示します。この枠線をアクションブロックに重ねることで、アクションブロックをクリックした場合と同じように表示色が変わります。このようにアクションブロックの表示色が変わった状態を「選択状態」といいます。選択状態のアクションブロックは、先ほど説明した場所の移動や削除・コピー・切り取り・貼り付けなどをまとめて行なうことができます。



7-5.矢印の経路の操作

矢印の途中にある丸印や、「はい」「いいえ」「A」「B」「戻る」などの文字の中心部分をクリックしてドラッグすると、矢印の折れ曲がる場所を変えることができます。矢印とアクションブロックが重なって見づらい場合などは、この操作を活用すると非常にプログラムを見やすくすることができます。



7-6.メニュー・ツールバーの説明

本ソフトウェアに備わっているメニューの項目とツールバーのボタンの機能について説明します。なお、同一の機能を持つメニュー項目とツールバーのボタンは、項目名の後に括弧でツールバーのアイコンを表示しています。

・ ファイル

- **新規作成**（）・・・・現在作成中のプログラムを破棄し、新しいプログラムの作成を開始します。
- **開く**（）・・・・・・・ファイルに保存したプログラムを読み込みます。
- **上書き保存**（）・・・現在作成中のプログラムを上書き保存します。ファイル名の無いプログラムの場合は名前を付けて保存を行ないます。
- **名前をつけて保存**・・現在作成中のプログラムを、名前を変更して保存します。
- **終了**・・・・・・・・・本ソフトウェアを終了します。

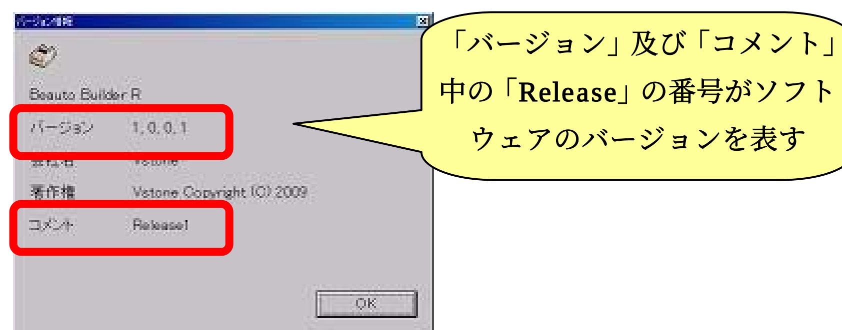
・ 編集

- **元に戻す**（）・・・・・現在作成中のプログラムをひとつ前の状態に戻します。
- **やり直し**（）・・・・・「元に戻す」で戻したプログラムの状態を一つ進めます。
- **切り取り**（）・・・・・現在プログラムエリアで選択しているアクションブロックをコピーして削除します。
- **コピー**（）・・・・・・現在プログラムエリアで選択しているアクションブロックをコピーします。
- **貼り付け**（）・・・・・「切り取り」及び「コピー」でコピーしたアクションブロックをプログラムエリアに貼り付けます。
- **削除**・・・・・・・・・現在プログラムエリアで選択しているアクションブロックを削除します。
- **すべて選択**・・・・・現在プログラムエリアに存在するすべてのアクションブロックを選択状態にします。

- <u>プログラム</u>
  - **プログラムの書き込み** ()・・・現在作成中のプログラムをロボットに書き込みます。
  - **プログラムの読み込み**・・・・・ロボットに記録されているプログラムを読み込んでプログラムエリアに復元します。
  - **プログラムのスクリーンショットを保存**・・・プログラムエリアの内容を 1 枚の画像ファイルに保存します。

- <u>設定</u>
  - **モーターの設定**・・・・・・・・・ロボット本体の車輪のスピードを調整します。
    上級者向け機能の使用設定・・・上級者向けの機能を使用するかどうかを設定するダイアログを開きます
  - **メモリマップの表示**・・・・・・メモリマップを画面に表示します

- <u>ヘルプ</u>
  - **バージョン情報**・・・・・本ソフトウェアのコピーライト表示、及びバージョン情報を確認します。この項目をクリックすると、下記のダイアログを表示します。



  - **ファームウェア書き換え**・・ロボット本体に書き込まれたプログラムを更新します。製品のサポートページで新しいバージョンのファームウェアが公開され、お使いのロボットをそのプログラムに書き換える場合に使用します。

7-7.ショートカットキーの説明

PC のキーボードより特定のキー入力を行なうことで、本説明書で紹介しているメニューの一部の項目を呼び出すことができます。この機能をショートカットキーといいます。なお、注意として、ショートカットキーを入力したときのマウスカーソルの位置が「プログラムエリア側か」もしくは「アイコンエリア・センサエリア・設定エリア側か」によって、操作内容が変わります。詳しくは下記をご参照ください。

●マウスカーソルがプログラムエリア側にある場合

- **Ctrl（コントロールキー）+A・・・メニューの「編集」→「すべて選択」と同じ**
- **Ctrl + C・・・・・・メニューの「編集」→「コピー」と同じ**
- **Ctrl + V・・・・・・メニューの「編集」→「貼り付け」と同じ**
- **Ctrl + X・・・・・・メニューの「編集」→「切り取り」と同じ**
- **Delete キー・・・・メニューの「編集」→「削除」と同じ**
- **Ctrl + Q・・・・・・メニューの「ファイル」→「終了」と同じ**
- **Ctrl + Z・・・・・・メニューの「編集」→「元に戻す」と同じ**
- **Ctrl + Y・・・・・・メニューの「編集」→「やり直し」と同じ**

●マウスカーソルがアイコンエリア・センサエリア・設定エリア側にある場合

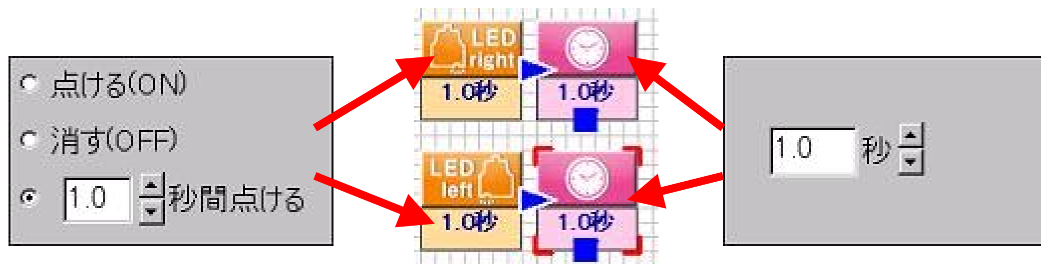
- **Ctrl + A・・・・・・数値設定のテキストを全て選択する**
- **Ctrl + C・・・・・・選択した数値設定のテキストを全て選択する**
- **Ctrl + V・・・・・・コピーしたテキストを数値設定項目に貼り付ける**
- **Ctrl + X・・・・・・選択した数値設定のテキストを切り取る**
- **Ctrl + Z・・・・・・数値設定のテキストを一つ前の入力状態に戻す**

8.その他

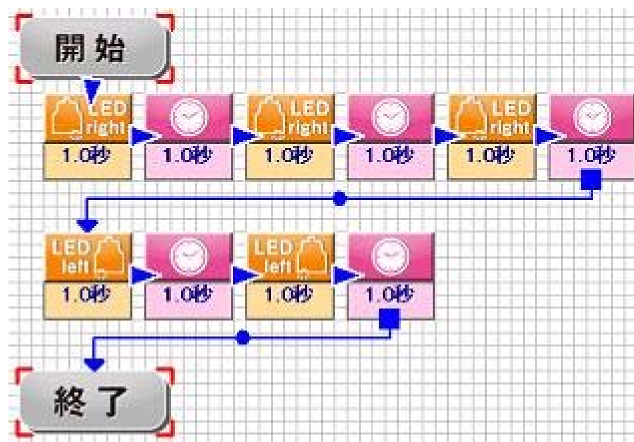
8-1.課題の答え

・　1 秒間隔で右 LED を 2 回、左 LED を 3 回点滅させるプログラム

　まず、左右の LED で「1 秒間で 1 回点滅する」というプログラムを作成します。使用する命令やその設定は下図の通りです。

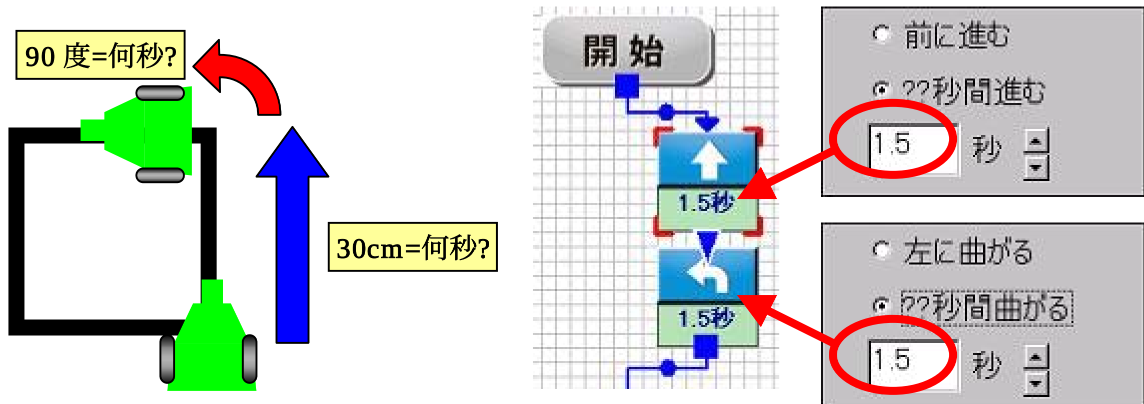


　続いて、右 LED の点滅を合計 3 つ、左 LED の点滅を合計 2 つ作成し、それぞれを下図の通りに接続します。これでプログラムは完成です。

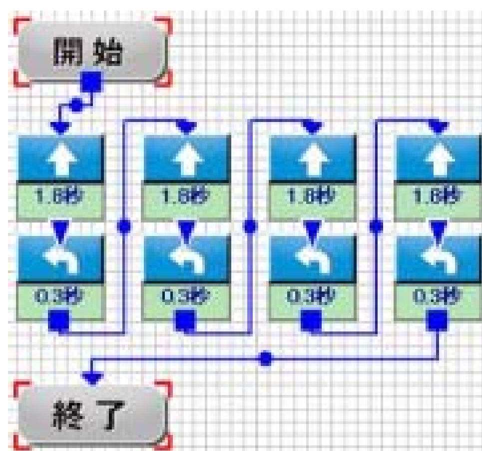


・　1 辺の長さが 30cm の正方形を紙に書き、それを一周なぞるプログラム

　まず、現在使用しているロボットについて「何秒間前進させると 30cm 進むか」及び「ない秒間旋回させると 90 度曲がるか」を確認します。下図のようなプログラムを作成し、何度も秒数を変更してプログラムを実行し、正しい秒数を調べてください。

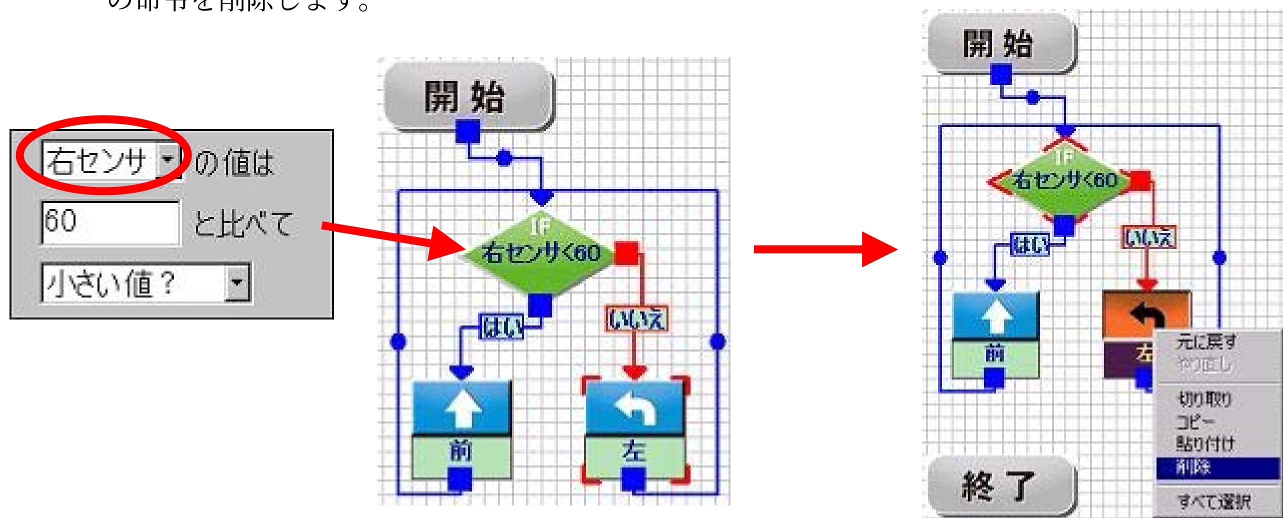


30cm の前進と 90 度の旋回ができたら、下図のようにそれを 4 つ並べて接続すればプログラムは完成です。ただし、微妙な誤差などによりまだうまく動作しない場合があるので、その場合は秒数を調整してください。

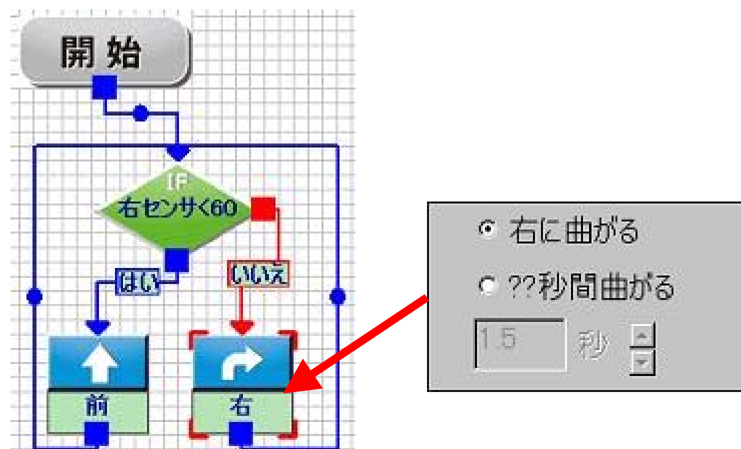


・ 右センサを使用してライントレースするプログラム

まず、分岐の命令より「左センサ」の設定を「右センサ」に変更します。また、「左旋回」の命令を削除します。



削除した左旋回の代わりに「右旋回」を追加し、下図の通りに接続したら完成です。なお、右旋回の命令は「右に曲がる」の設定を選択してください。



8-2.ロボット本体からのプログラムの読み込み

本ソフトウェアでは、ロボット本体に書き込まれたプログラムを読み込んでプログラムエリアに復元することができます。ロボット本体からプログラムを読み込む場合は、メニューより「プログラム」→「プログラムの読み込み」をクリックしてください。クリックすると、現在編集しているプログラムを保存するかどうか、また、読み込みを行なうかどうかの確認メッセージが表示されるので、読み込んでも問題ない場合は「はい」をクリックしてください。プログラムを読み込み終えると、プログラムエリアに読み込んだプログラムを表示します。また、左右車輪の速度も併せて読み込まれ、設定が更新されます。

なお、プログラムの読み込みについては、書き込んだときとまったく同じ内容を復元することはできず、それ以外にも下記の注意事項があります。読み込みを実行する際には十分注意してください。

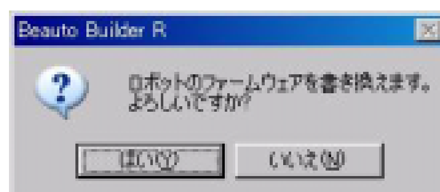
- アクションブロックの座標情報はロボットに記録されないため、復元されたプログラムでのアクションブロックの座標は書き込み時と大きく異なる場合があります。
- どこにも矢印を接続しなかったアクションブロックは、必ず「終了」に矢印がつながるようになります。
- 移動や LED のアクションブロックなどでは、時間の指定を行わなかった場合は秒数の設定自体がロボットに書き込まれないため、秒数の設定が書き込み時と異なる場合があります。
- 分岐や演算のアクションブロックでは、2 行目の項目に定数かメモリマップを選択しますが、選択されなかった方の情報はロボットに書き込まれないため、定数やメモリマップの設定が書き込み時と異なる場合があります。
- くり返しのアクションブロックで「ずっとくり返す(無限)」設定をしていた場合、読み込んだプログラムでは「ずっとくり返す(無限)」設定になりませんが、くり返しが終わった後に再びくり返しの開始に矢印がつながる仕組みになります（実質「ずっとくり返す(無限)」設定になります）。
- アクションブロックのつながり方や設定によっては、複数のアクションブロックが一つにまとめられる場合があります
- 速度や旋回量を指定できる移動のアクションブロックでは、設定によって「前進」「左右に曲がる」などに置き換わる場合があります。
- 上記のような差異は存在しますが、読み込んだプログラムを再びロボットに書き込んでも、まったく同じプログラムが実行されます。

8-3.ロボットのファームウェアの書き換え

ロボット本体には、PC と通信したり作成したプログラムを動かしたりするために、「ファームウェア」と呼ばれる最も基本的なプログラムが書き込まれています。ファームウェアは本ソフトウェアからプログラムを書き込んでも消えません。今後、ロボット本体のデバッグや機能の改善などによりファームウェアが更新された場合、新しいファームウェアをサポートページからダウンロードして書き込むことで、変更された点をお使いのロボットにも反映させられます。

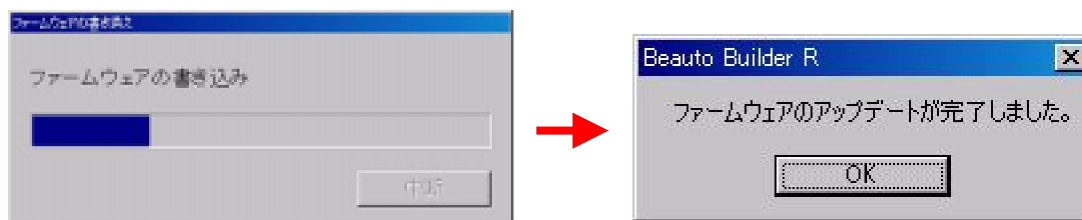
ロボット本体のファームウェアを書き換える場合は、以下の手順で行います。

まず、PC とロボットを接続して、メニューより「ヘルプ」→「ファームウェア書き換え」をクリックします。クリックすると、本当に書き換えを行なうかを確認するメッセージを表示するので、書き換えても問題ない場合は「はい」をクリックしてください。



「はい」をクリックすると、ファームウェアのファイルを選択する画面を表示するので、ダウンロードしたファームウェアのファイルを選択して「開く」をクリックしてください。

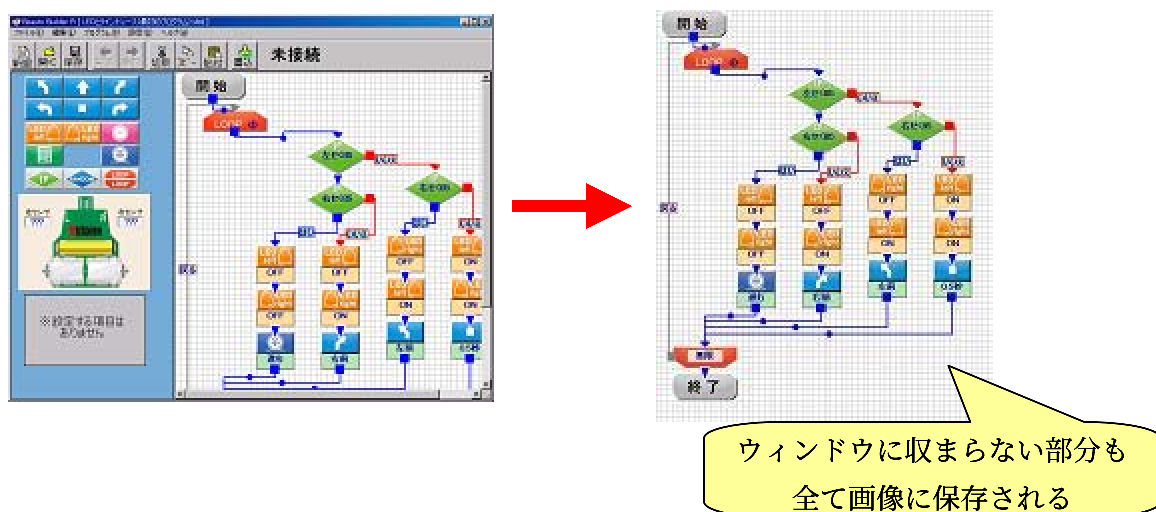
ファイルを開くと自動的にファームウェアの書き換えが始まります。プログラムの書き込みと同じように、書き込みが終了するまで待ってください。右下図のメッセージが表示されたらファームウェアの書き換えは終了です。



ファームウェアの書き換えに失敗した場合は、改めて書き込みを実行してください。

8-4.プログラムエリアの内容を画像に保存

メニューの「プログラム」→「プログラムのスクリーンショットを保存」をクリックすると、現在のプログラムエリアの内容を 1 枚の画像データに保存できます。画像は JPEG 形式で、画面に収まらない部分を含めた全ての内容が 1 枚の画像データに記録されるため、作成したプログラムに関する説明資料を作成する場合などに利用すると便利です。



メニュー項目をクリックすると、保存するファイル名を設定する画面を開くので、ファイル名を指定して「保存」をクリックしてください。クリックすると指定したファイル名で画像が作成されます。



なお、アクションブロックの選択状態など、実際に現在表示されている状態で画像が保存されます。また、画面外にはみ出している矢印は、他のアクションブロックと大きく離れていると画像内に収まらない場合がありますのでご注意ください。

8-5.Q&A

本ソフトウェアを使用して何か問題が発生した場合は、下記項目をご参照いただき、原因と対処方法をご確認ください。記載されている対処を行っても状況が改善しない、または以下に当てはまる現象が見つからない場合は、お手数ですが末尾の連絡先までお問い合わせください。

Q:ロボット本体の車輪がまわらない。

A1:ロボットの電池が少ない可能性があります。車輪の回転が弱々しい場合はこれが原因である可能性が高いです。電池を新しいものと交換して改善するかご確認ください。

A2:ロボットと PC を接続している場合、プログラムが実行されない場合があります。PC にロボットを接続していたら、PC からロボットを外してプログラムを実行してください。

A3:モーターの設定で、車輪の速度が非常に遅い設定になっている可能性が考えられます。「*3.モーターの設定*」の項目を参考に、車輪の速度を速く設定してください。

A4:移動のアクションブロックで、車輪を回す時間を非常に短く設定しているため、車輪が十分加速する前に停止してしまっている可能性があります。時間の指定を長めに設定しなおしてください。

A5:片方の車輪が何かに引っかかったりするなどして強い負荷がかかっている場合、ロボット全体の電力が不足して車輪が回らなくなる場合があります。車輪が引っかかっている場所などが無いかご確認ください。

A6:モーターをロボット本体に取り付ける向きや、部品の挿しこみが正しく行なわれていないと、モーターに電力が伝わらない場合があります。これらの点を一度ご確認ください。

Q:プログラムの書き込みの際に「ロボットと再接続しています」というメッセージが表示され続け、書き込みに失敗する

A1:ロボットを PC に何度か抜き差ししていると、PC の環境によって少しロボットを認識するまでの時間がかかる場合があります。一度ロボットを PC から抜き差するか、PC を再起動して書き込みを行なってください。

A2:ロボットにプログラムを書き込んでいる最中にロボットと通信が途切れた場合、このような状態になります。改めてロボットを PC にしっかり接続し、ロボットに触れたり動かしたりせずに書き込みを行なってください。

Q:プログラムを実行していないのに、勝手にモーターが動く

A:ロボットが PC と接続された状態で、ロボットの電源スイッチが ON になっていると、プログラムが急に始まるなどの問題が発生し、モーターが動く場合があります。ロボットを PC に接続する場合は必ず電源を OFF にしてください。

Q:ロボット本体の動作中に、頻繁に電源が落ちたりリセットがかかったりする

A1:ロボット本体の電池が少ない可能性があります。新しい電池に交換して状況が改善するかご確認ください。

A2: ロボットの車輪に無理な力がかかっており、全体の電力が不足している可能性があります。車輪が引っかかっているなどの問題が無いかご確認ください。

Q:ロボット本体から煙・火花・異臭などが発生した

A1:ロボットの配線がショートしている可能性があります。すぐに電源スイッチを OFF にして通信ケーブルを取り外し、電池を抜いた上で、末尾の宛先までご連絡ください。

Q:PC にロボットを接続しても通信が行なわれない

A1:ロボットを接続する向きが逆、奥まで差し込まれていない、斜めから差し込まれている、などの場合、ロボットと PC が正しく接触できず通信できない可能性があります。これらの点について一度ご確認ください。

A2:ロボットへのプログラムの書き込みが失敗したり、何らかの原因によって本ソフトウェアが強制的に終了したりした場合、PC がロボットを正しく認識できなくなる場合があります。この場合、一度ロボットを PC から取り外して再び取り付けることで改善する場合があります。これについて一度ご確認ください。

Q:PC とロボットの通信が急に途切れることがある

A1:PC にロボットを接続した状態で激しくロボットを動かるなどをすると、ロボットと PC の接触が不安定になり、途中で通信が途切れる可能性があります。PC にロボットを接続している場合は、余りロボットを激しく動かさないようにしてください。

A2:ロボットが奥まで差し込まれていない、斜めから差し込まれている、などの場合、ロボットと PC の接触が不安定になり、途中で通信が途切れる可能性があります。これらの点について一度ご確認ください。

Q:プログラムの書き込みの際に「プログラムデータのサイズが大きすぎます」というエラーが表示されプログラムを書き込むことができない

A:作成したプログラムがプログラムに書き込めるデータのサイズを越えた場合、ロボット本体にプログラムを書き込むことができません。プログラム中に存在する矢印の接続されていないアクションブロックもロボット本体にデータが書き込まれるので、不要なブロックは削除して、プログラムを書き込めるサイズまで容量を削減してください。

Q:演算命令でメモリマップの数値が意図したものに書き変わらない

A1:メモリマップの各項目は 0~255 の整数しか扱うことができないため､演算の結果がこれらの数値になる場合は、数値がこの範囲内の値に自動的に補正されます。演算の結果が 256 以上の場合は 255 に、また、負の数の場合は 0 に修正されます。また、割り算の場合は小数点以下の数値が切り捨てられます。

Q:本ソフトウェアを実行すると「gdiplus.dll が指定されたパスに見つかりません・・・」というエラーが表示され、本ソフトウェアを利用できない。

A:Windows2000、及び古い PC など、PC に gdiplus.dll というファイルがインストールされていない場合、このエラーが発生します。この場合、本ソフトウェアの「gdilpus」というフォルダに入っている「gdilpus.dll」というファイルを、本ソフトウェアの実行ファイル「cl_edit_r.exe」と同じフォルダにコピーしてください。

Q:シミュレータで仮想ロボットが動かない。

A3:シミュレータのモーターの設定で、車輪の速度が非常に遅い設定になっている可能性が考えられます。「シミュレータのモーターの設定」の項目を参考に、車輪の速度を速く設定してください。

A4:移動のアクションブロックで、車輪を回す時間を非常に短く設定しているため、車輪が十分加速する前に停止してしまっている可能性があります。時間の指定を長めに設定しなおしてください。

Q:シミュレータで仮想ロボットがゴールに到着したのにゴールと認識されない。

A:一部のコースでは、ゴールを通過するのではなくゴールの中でしっかり停止しないとゴールしたと判断されません。

Q:シミュレータで仮想ロボットがコースから外れていないのに「コースを通る順番が違います」と表示される

A:一部のコースでは、ラインから外れたりしなくても、コース上の決められた場所を決められた順番で通過する必要があります。この順番が逆だったり、全ての場所を通過せずゴールしたりすると、正しくコースを走ったとみなされません。

●お問合せ先

ヴイストン株式会社

〒554-0012 大阪市西淀川区御幣島 2-15-28

Tel:06-4808-8701 Fax:06-4808-8702

e-mail: infodesk@vstone.co.jp

製品サポート URL: http://www.vstone.co.jp/robot/beautoracer/rdownload.html

URL: http://www.vstone.co.jp/

(2010/06/25)